3 Web Designs in 3 Weeks

@juntao.qiu

Table of Contents

前言

关于3周3页面

2014年11月，我在ThoughtWorks西安办公室组织了一次为期3周的Workshop，名为《3周3页面》。每一周的两次课上，参与者被要求使用HTML/CSS来实现一个Web页面。

这个Workshop有点像学绘画时候的素描课，所有人都照着一个设计稿（通常是设计师用PhotoShop等工具设计好的一个页面）来用HTML/CSS实现它。

参与者有前端开发，非前端开发，测试工程师，用户体验设计师，业务分析师等角色。大部分参与者在之前都或多或少的接触过CSS，但是又没有到精通的地步；有很小一部分参与者之前没有接触过Web设计/开发，同样，有很小一部分参与者在Web设计/开发方面非常专业。

在Workshop中，我会在每周的第一次课上讲解一些基础知识，比如如何使用HTML5的新标签来实现更加语义化的文档，如何使用CSS3的新特性来实现阴影，动画，圆角等。而在第二次课，会安排一次Showcase，大家将自己已经实现了的部分与别人分享，并得到一些反馈。在Showcase之后，我会现场演示如何实现页面的一部分，这样可以帮助那些并没有太多基础的参与者。

结果这个Workshop得到了很好的反馈，参与者都投入了很大的经历和时间来完成“作业”，到3周结束的时候，绝大多数人都有了基本的概念，并且真的可以照着一个设计稿来完成Web开发部分的工作。而且根据课后的一次问卷调查的结果来看，大部分参与者非常乐于这种边讲边练的模式，并且期望学到更多。

在进行自我总结的时候，我突然意识到，这是一个很容易固化下来，方便其他没有参加这个Workshop的人的好主题。于是我决定将这些内容整理起来，做成一个电子书，这样也可以省去我自己重复去讲课的麻烦。

一些更新

《3周3页面》发布在gitbook上之后，出乎意料的获得了很多来自各方的支持，在此一并感谢。有一次在ThoughtWorks内部邮件里，我向同事们宣布了《3周3页面》已经变成了一本电子书的消息，结果收到了很多的捐赠，ThoughtWorks同事们对创新，知识分享的支持和热情都令我非常感动。

在深圳出差的时候，selfstore.io的创建者黄增光联系我，希望我可以将本书的PDF版本发布在selfstore.io上。发布之后，已经有很多读者购买，在这里感谢黄增光和热心的读者们。

上一个项目结束之后，我难得的有了几天空闲，就将上一个项目中涉及到的CSS3动画知识整理了出来，形成了新的一章。动画一章中，实例部分的动画来自于ThoughtWorks的用户体验设计师唐婉莹，她不但慷慨的允许我在书中使用这些设计稿，而且还教了我很多动画效果的实际做法；实现动画时，在处理多个元素的动画同步上我遇到了难题，同事张霄翀给了我热心的的指导，并帮我做出了一个样例，在此一并感谢。

最后，在本书发布之后，很多同事都帮我做了积极的宣传，使得本书得到了更多的人的关注，这里也一并感谢。

第一周：Find An Agent

第一周里，我们要实现一个`Find an agent`的页面，这是一个非常典型的，现代的Web设计稿。我们会分析如何分解页面，用标记语言编写，然后加入样式来实现它。不过在开始之前，我们需要先熟悉一些基础知识。

第一天

在第一天里，我们会学习一些基本的Web开发知识，以及一些工具的使用，然后就开始第一个页面`Find an agent`的实际开发。

用户界面

优秀的程序员都偏爱命令行界面，文本格式的协议，这样一切都是简单而可读的，比如我们每天工作的Shell环境：

```
➜  web-starter-kit-master git:(master) ✗ gulp serve
[01:01:39] Using gulpfile ~/develop/design/web-starter-kit-master/gulpfile.js
[01:01:39] Starting 'styles'...
[01:01:41] gulp-ruby-sass: directory
[01:01:42] gulp-ruby-sass: write components.css
[01:01:42] gulp-ruby-sass: write components.css.map
[01:01:43] gulp-ruby-sass: write main.css
[01:01:43] gulp-ruby-sass: write main.css.map
[01:01:45] 'styles' all files 185.18 kB
[01:01:45] Finished 'styles' after 5.72 s
[01:01:45] Starting 'serve'...
[01:01:45] Finished 'serve' after 80 ms
[BS] Local URL: http://localhost:3000
[BS] External URL: http://192.168.0.104:3000
[BS] Serving files from: .tmp
[BS] Serving files from: app
```

但是对于最终用户（比如在淘宝上购物的程序员）来说，这种界面是非常糟糕的。所有的信息都展现出来了，但是没有优先级。也就是说，强调一切等于什么都不强调。

我们再来看一个例子

```
***<没钱赚商店>购物清单***
名称：雪碧，数量：2瓶，单价：3.00(元)，小计：6.00(元)
名称：可口可乐，数量：4瓶，单价：15.00(元)，小计：60.00(元)
----------------------
挥泪赠送商品：
名称：雪碧，数量：1瓶
----------------------
总计：60.00(元)
节省：3.00(元)
**********************
```

对于程序员来说，这样的`用户界面`非常熟悉。但是我们可以做的更好：

没钱赚商店

商品清单

雪碧
数量：2
可口可乐公司出品

可口可乐
数量：2
可口可乐公司出品
NEW

可口可乐
数量：2
可口可乐公司出品
NEW

挥泪赠送

雪碧
数量：1
可口可乐公司出品

总计：60.00

节省：3.00

同样的信息，以不同的方式展现之后，对于信息的获得者来说是天差地别的。一个好的用户界面，可以更容易的将信息传递出来，使得使用者的生活变得更简单一些。

一些基础知识

好了，我们已经明白了用户界面的重要性。在接下来的章节中，我们讲学习如何实现一个好的用户界面。首先我们来学习一些接下来要使用的工具，以及一些现代的工作方式。

Sass和Compass

Sass是一个CSS3的扩展语言，它提供了丰富的特性使得编写样式更加容易：嵌套样式，变量定义，扩展，mixin等等。一旦你开始使用它，你就再也回不去了。

我们可以看几个简单的例子来了解Sass的基本语法，比如在Sass中，可以很方便的定义变量：

```
$body-color: #efefef;
$text-color: #4f4f4f;

body {
    background-color: $body-color;
    color: $text-color;
}
```

```
a:hover {
    background-color: $text-color;
    color: $body-color;
}
```

另外一个非常好用的功能是嵌套定义：

```
$highlight-color: lightgreen;

.hero {
    h1 {
        font-size: 4em;
    }

    p {
        color: white;
        a {
            color: $highlight-color;
        }
    }
}
```

上边的代码经过预处理，会生成这样的 `css` 代码：

```
.hero h1 {
  font-size: 4em;
}

.hero p {
  color: white;
}

.hero p a {
  color: lightgreen;
}
```

稍加比较，就会发现Sass的语法比纯粹的CSS要简洁很多，并且可以很容易的看出层次关系。

另外一个非常 `高级` 的特性是 `mixin`。如果你熟悉Ruby的话，那么可以很容易的将Ruby中的经验用到这里来。简而言之，`mixin` 机制就是将一段独立的，可复用的代码 `植入` 到当前的代码中。

```
@mixin radius($radius: 5px) {
    border-radius: $radius;
}

.hero {
    @include radius(10px);
}

.gallery {
    p {
```

```
            background-color: $body-color;
            @include radius(5px);
        }
}
```

编译之后，会生成这样的 `css`：

```
.hero {
  border-radius: 10px;
}

.gallery p {
  background-color: #efefef;
  border-radius: 5px;
}
```

关于Sass就暂时介绍这么多，我们在后边用到的时候再做讨论。接下来是 `Compass` 这个工具了。简而言之， `Compass` 是一个使用了Sass的库，它将很多常用样式打包成了一些模块，我们可以在自己的项目中使用这些模块，比如模块 `reset` 可以用来将不同浏览器的差异抹平， `css3` 则用来生成 `CSS3` 相关的属性等。

安装Compass非常容易：

```
$ gem install compass
```

安装之后，就可以使用命令行工具 `compass` 来创建Sass工程了：

```
$ compass create my-project
```

这条命令会在当前目录下生成 `my-project` 目录，并在其中生成一些配置文件和相关的目录结构：

```
$ tree my-project
my-project
├── config.rb
├── sass
│   ├── ie.scss
│   ├── print.scss
│   └── screen.scss
└── stylesheets
    ├── ie.css
    ├── print.css
    └── screen.css
```

`sass` 目录中就是对应的源文件目录，我们在该目录中进行代码的编写，而通过下列命令来进行编译：

```
$ compass compile
```

生成的css文件位于`stylesheets`目录中。

使用`Compass`，只需要引入它提供的模块，就可以生成良好的跨浏览器的`CSS`片段：

```
@import "compass/css3";

.hero {
    @include border-radius(10px);
}
```

这段代码会生成：

```
/* line 3, ../sass/nested-css.scss */
.hero {
  -moz-border-radius: 10px;
  -webkit-border-radius: 10px;
  border-radius: 10px;
}
```

Sublime编辑器

Sublime是一个非常好用，非常现代的编辑器。Sublime是一个跨平台的编辑器，你可以在Windows，Linux以及Mac OSX上使用它。虽然它不是免费的，但是如果你不购买，功能上没有任何的限制（除了不定时的弹出一个对话框外）。

虽然Sublime现在才仅仅是第3个版本，但是在程序员群体，特别是Web前端程序员这个群体中，Sublime的占用率已经非常高了。

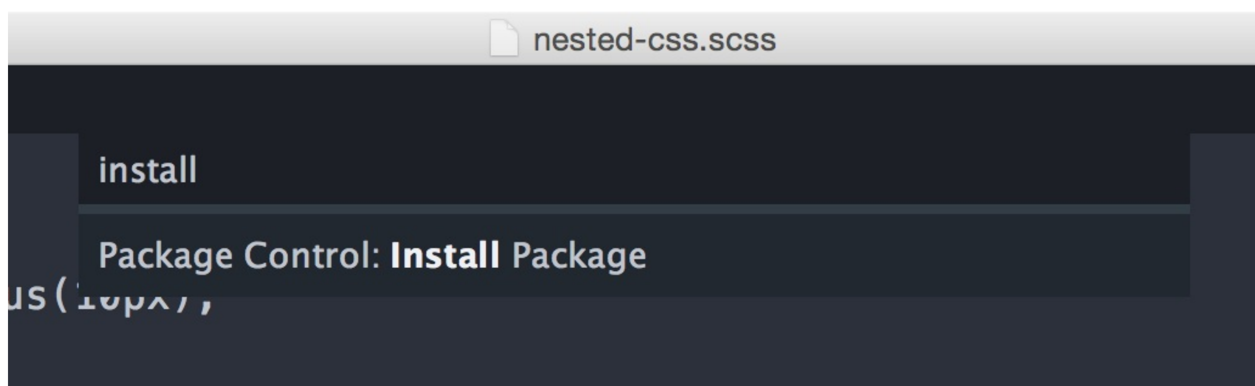
和众多优秀的编辑器一样，Sublime提供一个非常不错的插件机制，这样用户就可以自由扩展它了。

一旦你开始使用Sublime，首先需要安装的一个插件就是`管理其他插件的插件`：Package Control。请按照其官方站点的介绍安装。

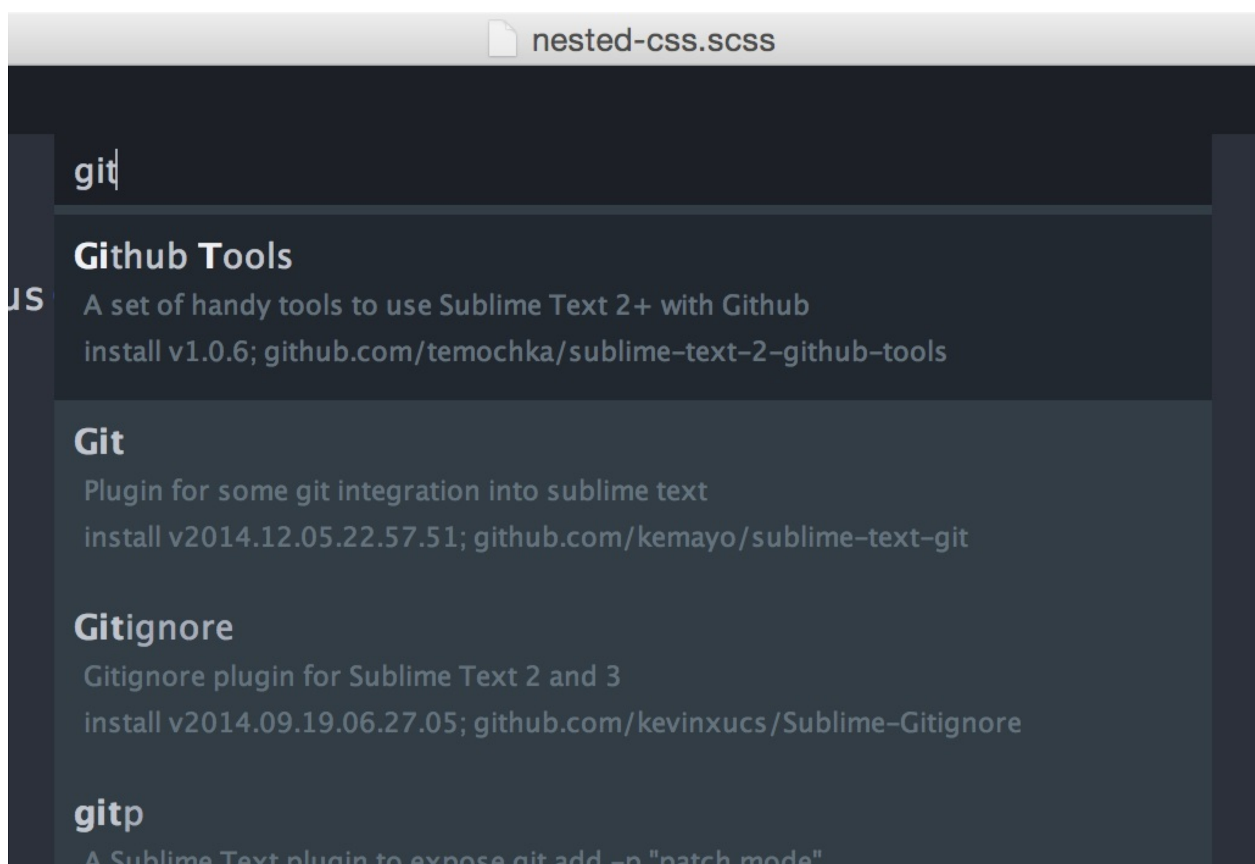
安装了`Package Control`之后，我们来安装几个非常好用的插件：

1. Emmet
2. HTML-CSS-JS Prettify
3. GitGutter

安装插件非常容易，`Command + Shift + P`打开命令窗口，然后输入`install package`：



然后输入要安装的插件名称，



Emmet是一个用于快速生成HTML片段的工具，他本身并不会独立存在，事实上它与编辑器一起工作才能施展出自己的能力。Emmet支持众多的编辑器，比如Sublime，Textmate，Notepad++等。

Emmet自己定义了一个类似CSS选择器的`语言`，你可以用这个`语言`快速的描述自己想要的HTML文档结构，然后触发一个快捷键，Emmet会自动生成对应的HTML片段。

我们来看几个小的例子。比如我们需要一个`section`元素，该元素中有一个`h2`元素，和3各`p`元素：

```
<section>
    <h2></h2>
    <p></p>
    <p></p>
    <p></p>
</section>
```

如果使用Emmet的语言来描述的话，我们会写出这样的表达式：

```
section>h2+p*3
```

其中 `>` 表示子一级的节点，而 `+` 表示兄弟节点，最后 `*` 表示重复，上边的语句可以读作："生成一个section，这个section下有一个h2，同时该h2有3个paragraph作为兄弟"。

在来看一个稍微复杂一点的场景，我们有一个长度为10的列表，列表中每个元素都是一个类为 `item` 的div，每一个div中，有一个h2元素，然后有一个图片，图片文件分别为 `image-1.jpg` 到 `image-10.jpg` 。

那么，我们写出来的Emmet表达式为：

```
ul>li*10>.item>h2+img[src="image-$.jpg"]
```

这样它会生成我们预期的片段：

```
<ul>
    <li>
        <div class="item">
            <h2></h2>
            <img src="image-1.jpg" alt="">
        </div>
    </li>
    <li>
        <div class="item">
            <h2></h2>
            <img src="image-2.jpg" alt="">
        </div>
    </li>
    ...
</ul>
```

`.item` 会生成一个class为 `item` 的div元素， `$` 表示迭代中的index，它会被动态的替换为 `1` 到 `10` 的数字。

`HTML-CSS-JS Prettify` 是一个用来格式化 `HTML` ， `CSS` 以及 `JavaScript` 文件的插件。在Mac OS X上，通过快捷键为 `Command + Shift + H` 来触发格式化的动作。

`GitGutter` 是一个用来在Sublime的侧边栏上显示当前文件在Git中的状态的插件，它可以图形化的展示那些内容有修改等：

```
 8
+ 9  Before you get started, please make sure you have `Ruby`
+10  you actually have a lot of chooses:
+11
+12  1. rvm.io
+13  2. rbenv
+14
 15  To use it, simply clone this repo and :
 16
```

静态服务器

使用HTML/CSS开发页面的一个好处就是一切都是静态的，如果只是在本地开发，我们甚至都不需要任何的服务器：在浏览器中打开HTML文件即可。

但是一个好的实践是在本地运行一个简单的HTTP服务器。一方面，这个服务器可以很容易的将我们的成果展示给其他人，另一方面，可以避免一些不必要的麻烦：比如JavaScript访问服务器上的资源时，本地文件协议`file://`会和`http://`协议的请求（比如ajax）发生跨域的限制。

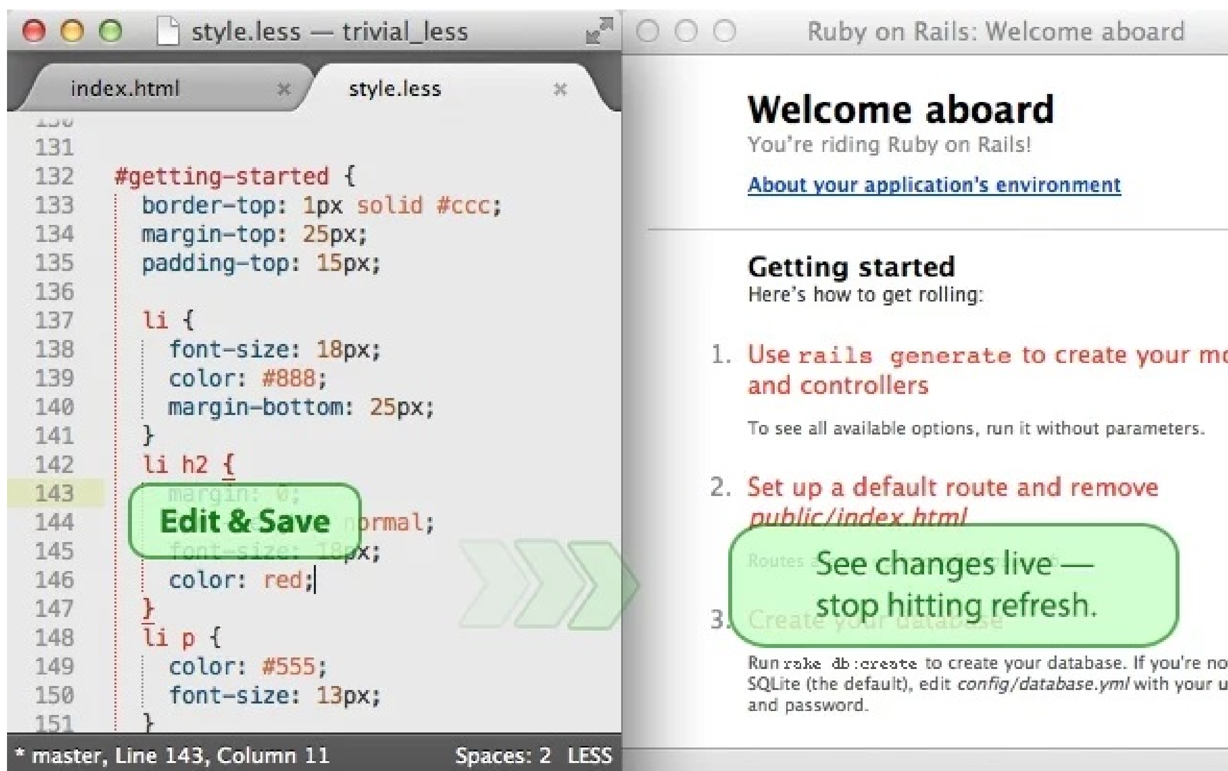
Mac OS X自带了`python`，因此可以很容易的使用`python`自带的HTTP服务器来：

```
$ python -m SimpleHTTPServer 9999
```

这条命令会把`运行该命令的目录`变成一个HTTP的服务器的根目录，并在`9999`端口上运行。比如当前目录中包含了一个`index.html`，那么在当前目录运行该命令之后，就可以在浏览器中访问这个`index.html`文件了。

LiveReload

LiveReload是一个浏览器插件，它可以和后台的服务器相关的服务器通信，当服务器通知该插件后，该插件会自动刷新页面。



模板工程

方便起见，我已经设置了一个简单的模板工程，这个工程中包含了建立一个Web设计所需要的几乎所有的文件及配置。

这个工程有一下配置：

1. 一个HTML5的样板文件
2. Compass的配置文件（用以编译SCSS）
3. Guard的配置文件（用以自动刷新页面）

Guard是一个命令行工具，它可以检测到本地文件的修改。当我们知道文件被修改之后，就可以做一些有趣的事情。比如当HTML文件发生变化时，我们想要浏览器被自动刷新，或者当scss文件发生修改时，我们需要触发Compass进行一次编译。

其中Guardfile的配置如下：

```
guard 'livereload' do
  watch('index.html')
  watch(%r{stylesheets/.+\.(css)})
  watch(%r{scripts/.+\.(js)})
end

guard :compass
```

这个文件定义了两个规则。第一个是关于 `livereload` 的：如果当前文件夹中的 `index.html` 或者 `stylesheets/*.css` 或者 `scripts/*.js` 发生变化，都会触发 `livereload` 这个任务。第二个是 `compass` 的，即当 `sass` 目录中的 `*.scss` 文件发生变化时， `compass` 会启动编译的动作，这个动作会生成 `stylesheets` 目录下的 `*.css` ，然后这又会触发 `livereload` 。

环境设置

有了基础知识之后，我们来在本地设置开发环境，这个配置方式我们在后续的章节也会用到。

首先，将远程的模板工程下载到本地：

```
$ git clone git@github.com:abruzzi/design-boilerplate.git mydesign
```

然后在该目录中执行bundle install即可（当然，你需要先安装Ruby）

```
$ cd mydesign
$ bundle install
```

这样基本的安装就完成了！

开发流程

我通常会启动两个终端，一个用来运行Guard，另一个用来运行HTTP Server，最后启动浏览器。

在终端1中，输入：

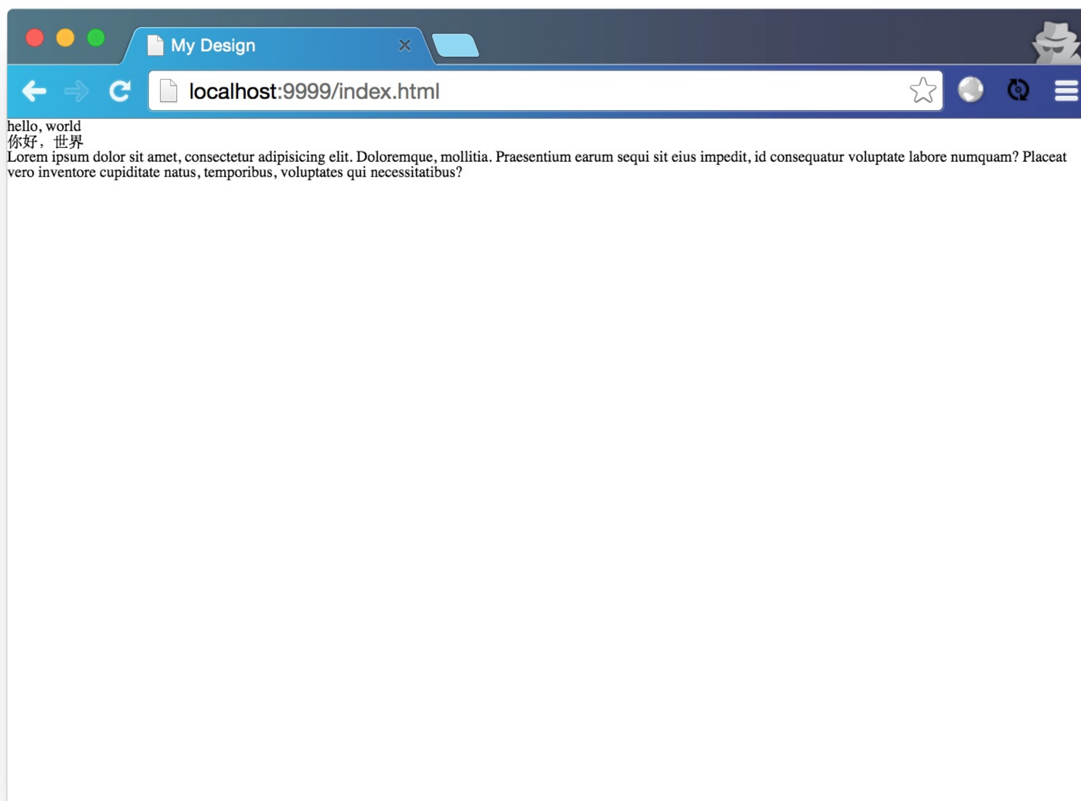
```
$ guard start
```

```
➜  mydesign git:(master) ✗ be guard
00:28:12 - INFO - Guard is using TerminalTitle to send notifications.
00:28:12 - INFO - Guard::Compass is waiting to compile your stylesheets.
00:28:12 - INFO - LiveReload is waiting for a browser to connect.
00:28:12 - INFO - Guard is now watching at '/Users/jtqiu/develop/design/mydesign'
[1] guard(main)>
```

来启动`Guard`的服务器，然后在终端2中，输入：

```
$ python -m SimpleHTTPServer 9999
```

启动HTTP服务器，这时候打开浏览器，输入`http://localhost:9999/index.html`应该可以看到：



这时候，在Sublime里打开整个工程，编辑 `index.html` 文件，结果浏览器中的页面并没有自动刷新，这是因为我们还没有连接浏览器和 `Guard` ，点击浏览器菜单栏的 `LiveReload` 的图标，你应该会在启动 `Guard` 的那个终端中看到这样的提示：

```
00:28:12 - INFO - LiveReload is waiting for a browser to connect.
00:28:12 - INFO - Guard is now watching at '/Users/jtqiu/develop/design/mydesign'
[1] guard(main)> 00:35:02 - INFO - Browser connected.
```

这样两者就连接起来了。这时候如果你修改 `index.html` 或者 `style.scss` 文件，页面都会自动刷新了！我们后续的开发都假设你已经完成了上述的设置。

第一个页面

我们这周选择的页面，来自于dribble，是一个 `寻找房产中介` 的站点的设计：


Agents
Login
Sign up
Relax, Finding An Agent Just Got Easier
Receive proposals from best agents for free
Enter a Suburb
SEARCH
20,000
Agents
1, 723,000
Transactions
17, 414
Agencies
10,200
Suburbs
HOW IT WORKS
1
2
3
Compare
Magni dolores eos qui ratione voluptatem sequi nesciunt neque porro quisquam est
Receive Proposals
Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat
Sell with the Best
Incidunt ut labore et dolore magnam aliquam quaerat voluptatem.
WHY CHOOSE US?
Quick
Nemo enim ipsam voluptatem quia voluptas sit aspernatur aut odit aut fugit, sed quia consequuntur magni dolores eos.
Free
Nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit.
Easy
Qui ratione voluptatem sequi nesciunt. Neque porro quisquam est, qui dolorem ipsum.
Independent
Onsectetur, adipisci velit, sed quia non numquam eius modi tempora incidunt ut labore et dolore magnam.
Comprehensive
Cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum. Sed ut perspiciatis unde.
Awesome
Sed ut perspiciatis unde omnis iste natus error sit voluptatem accusantium doloremque laudantium.
Testimonials
Recent Feedbacks
Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.
John Mayer
Halcyon, CEO
Find An Agent
Enter a Suburb
SEARCH
Blog
Contact Us
About Us
FAQs
Privacy
Disclaimer

这个设计使用了一个非常常见的布局方式：一个嵌入在hero图片上的导航，一个简单的搜索表单，然后一次是几个独立的块，每个块中的内容相互隔离，最后是搜索表单的一个重复，然后是页脚。

Mocking Up

根据我们初步对设计的解读，我们可以将页面分解成若干个小的部分：

1. Hero区域（包括导航，搜索表单）
2. 数字区域（罗列了若干个数字）
3. 工作方式区域（How It Works）
4. 优势区域（Why Choose Us）
5. 评论区域
6. 底部表单区域
7. 页脚

对一个页面的开发，通常来说有两种方式：

1. 先将整个页面的HTML编写出来，然后添加CSS样式化
2. 先实现某个区域的HTML，然后样式化，然后下一个区域

这两种开发方式各有利弊，但是又殊途同归，你可以选择适合自己的方式。我个人比较倾向于使用第2中方式。这种方式可以激励我进行下一步的工作，也可以比较明确的告诉我当前的进度是什么。

下面我们就来按照从上到下的次序来实现这个页面。

Hero区域

在现代的Web设计中，设计师会将一个巨大的图片设置在最醒目的位置，这张图片被称为 `Hero Image` 。 `Hero Image` 一般会和主题相关，但是有不至于喧宾夺主（通常会选择加上模糊路径或者色彩偏暗的图片）。

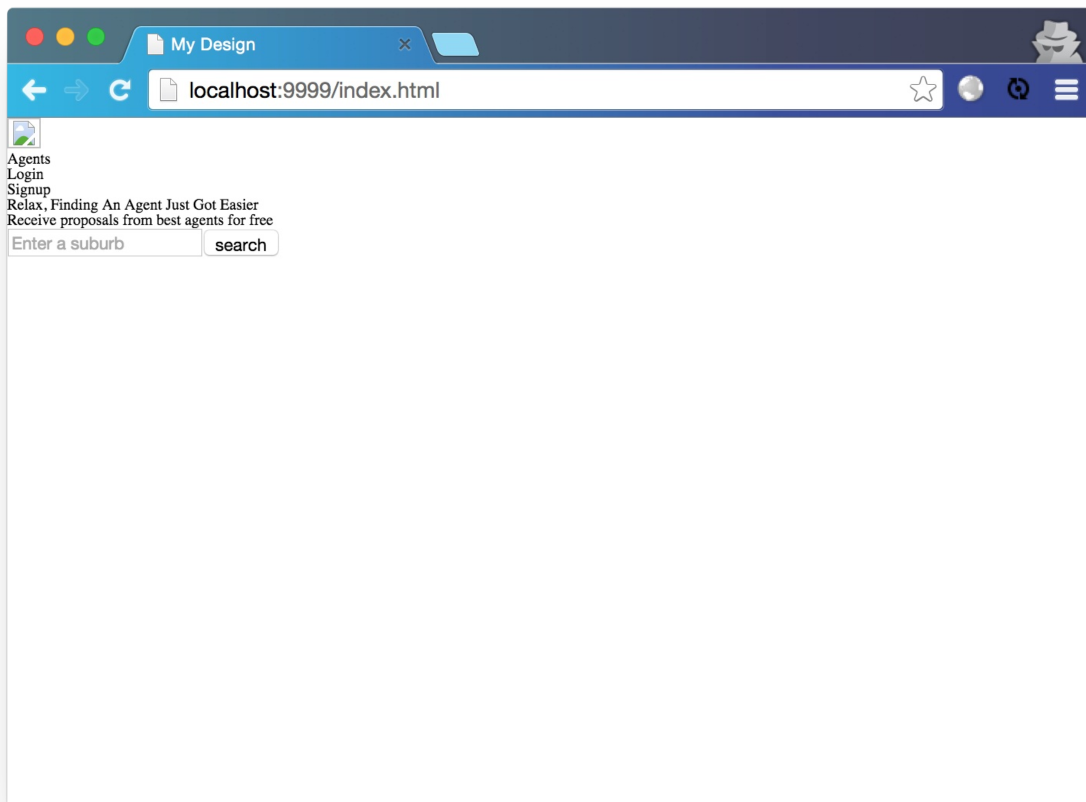
在Hero区域，我们可以看到一个logo，三个导航按钮，一个主标题，一个副标题，一个搜索表单。对应的，我们可以很直观的将其翻译为HTML文档：

```
<section class="hero">
    <header>
        <img src="images/logo.png" alt="logo">
        <ul>
            <li>Agents</li>
            <li>Login</li>
            <li>Signup</li>
        </ul>
    </header>
    <h1>Relax, Finding An Agent Just Got Easier</h1>
    <h2>Receive proposals from best agents for free</h2>
    <form action="">
        <input type="text" placeholder="Enter a suburb"><input type="button" value="searc
    </form>
</section>
```

对应的 `Emmet表达式` 如下：

```
section.hero>(header>img+ul>li*3)+h1+h2+(form>input[type="text"]+input[type="button"])
```

此时没有添加任何样式，我们看到的页面是这样的：



让我们来编写一个 `scss` 代码：

首先我们将所有的文本都居中对齐：

```
body {
    font-size: 62.5%;
    text-align: center;
}
```

然后将logo和导航都浮动起来，logo浮动在左侧，导航浮动在右侧：

```
header {
    img {
        width: 8em;
        height: 2em;
        float: left;
    }
```

```
    ul {
        float: right;

        li {
            float: left;
            margin: 0 .5em;
            padding: 1em;
        }
    }
    @include clearfix;
}
```

注意底部的`clearfix`这个`mixin`，这是对于浮动元素的一个清除，如果不做清除浮动的动作，那么`header`之后的元素会在布局上受到影响。

做完这些调整之后，我们的scss代码就成了：

```
.hero {
    max-width: 100em;
    margin: 0 auto;

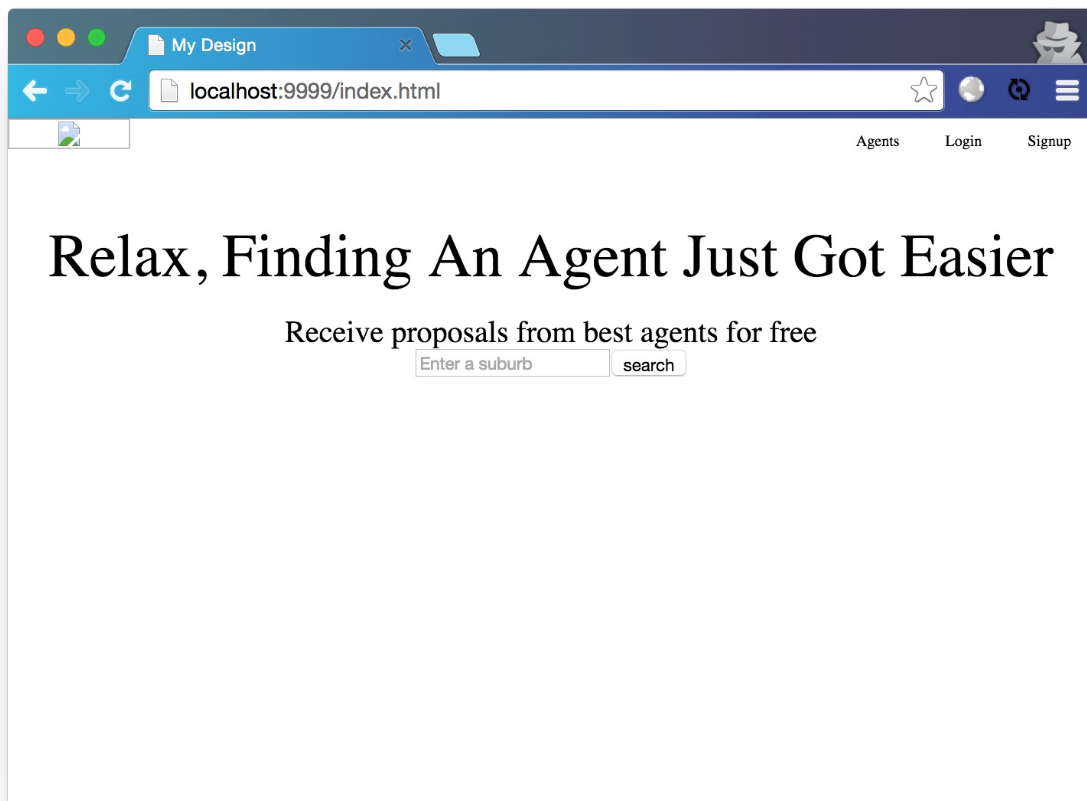
    header {
        img {
            width: 8em;
            height: 2em;
            float: left;
        }
        ul {
            float: right;

            li {
                float: left;
                margin: 0 .5em;
                padding: 1em;
            }
        }
        @include clearfix;
    }

    h1 {
        font-size: 4em;
        padding: 1em 0 .5em 0;
    }

    h2 {
        font-size: 2em;
    }
}
```

而对应的界面上的基本布局已经有了：



让我们为当前的设计找一张合适的主题图片。由于站点本身是与找房子相关的，因此我们可以找一些包含建筑，城市等主体的图片。

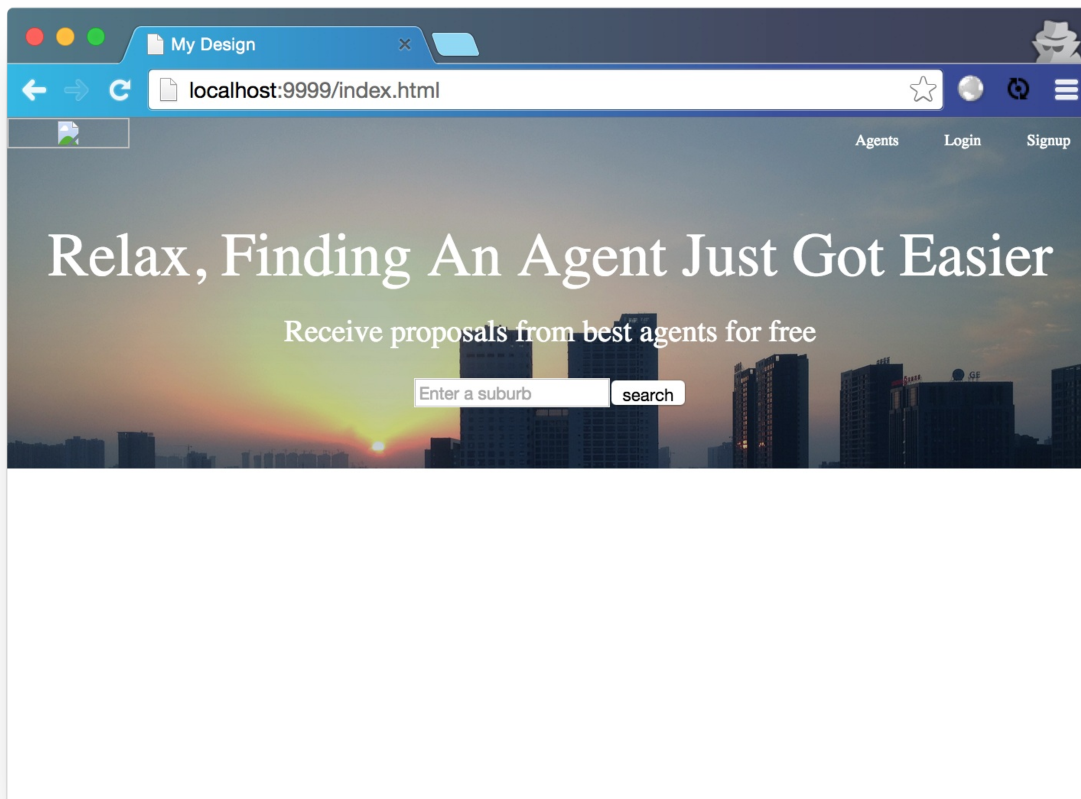
下面这张图片是我在ThoughtWorks西安办公室用手机拍摄的，我用PhotoShop做了一些简单的处理：



然后给 `hero` 添加背景图片，同时，由于背景颜色变深，因此我们采用白色作为文字颜色：

```
background: url('/images/find-an-agent-hero.png');
background-size: cover;

color: $highlight-text-color;
```



接下来，我们来样式化搜索表单，如果仔细观察，有一些细节需要处理:

1. 搜索框的背景为半透明
2. 搜索框的左上角和左下角有圆角
3. 按钮的右上角和右下角有圆角

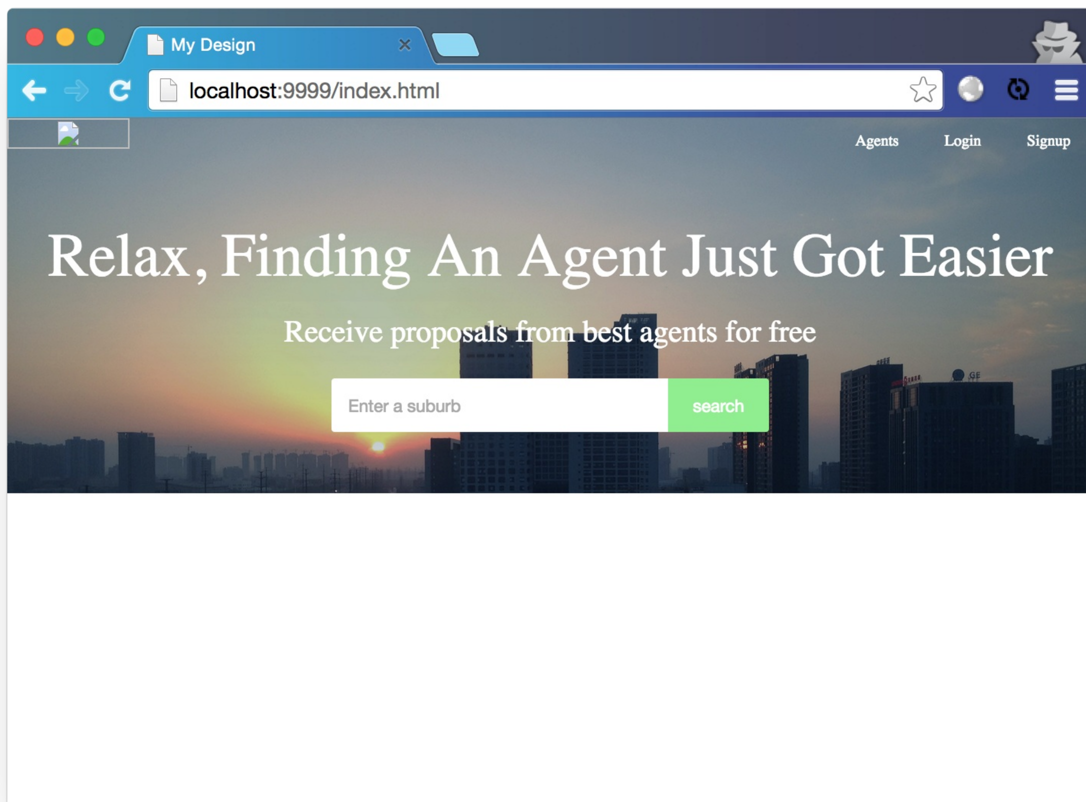
```
form {
    padding: 2em 0 4em 0;

    input[type="text"] {
        padding: 1em;
        border: 0;
        width: 18em;
        border-radius: 2px 0 0 2px;
        background: $bg-color;
        opacity: .3;
    }
```

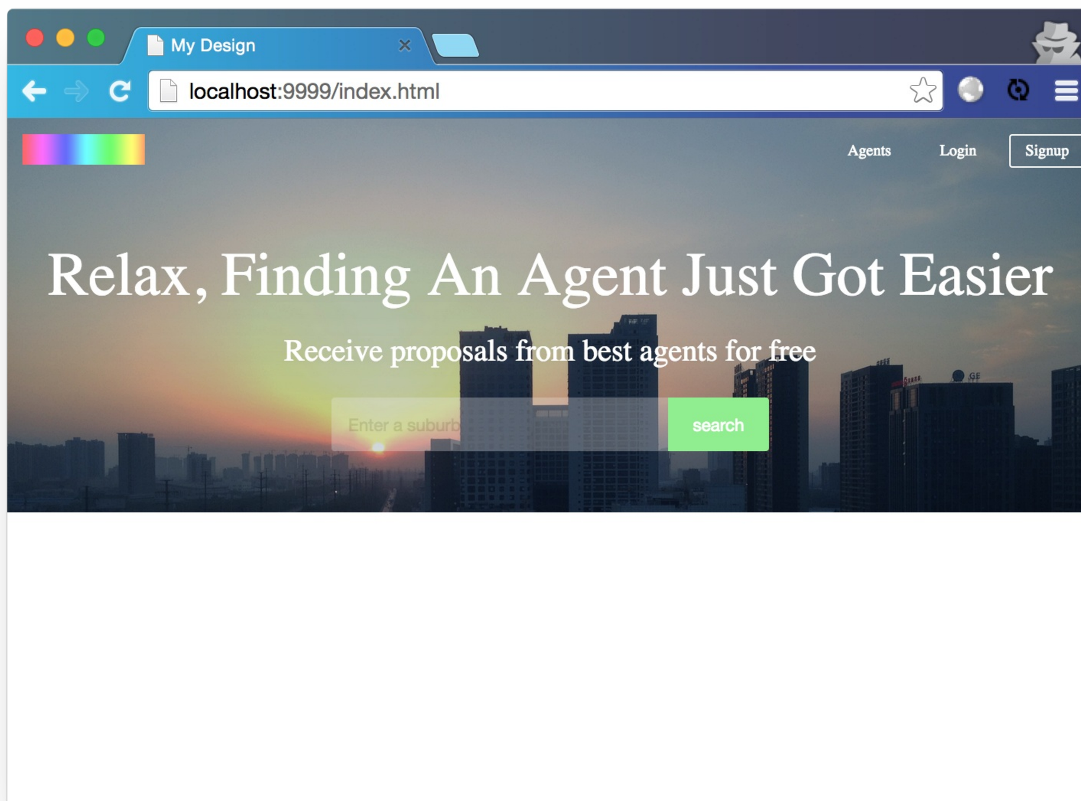
```
    input[type="button"] {
        color: $highlight-text-color;
        padding: 1em;
        border: 0;
        width: 6em;
        border-radius: 0 2px 2px 0;
        background-color: lightgreen;
    }
}
```

注意此处的 `border-radius` 属性，它可以分别制定四个角的圆角半径，次序分别为：左上，右上，右下，左下。对于按钮，我们可以看到设计稿上的颜色为淡绿色，这里先用 `lightgreen` 作为占位符，我们下一步会将其修改为真实的颜色。

现在的界面效果如下：



我们还需要一些细节需要处理，比如 `Sign up` 按钮的边框，logo等：



对于导航，我们先为每一个 `li` 都设置了透明的1个像素的边框，然后对最后一个，也就是 `Signup` 按钮，设置了1个像素的白色边框，同时设置了半径为2个像素的圆角。

```
li {
    float: left;
    margin: 1em .5em;
    padding: .5em 1em;
    border: 1px solid transparent;
    border-radius: 2px;

    &:last-child {
        border: 1px solid $highlight-text-color;
    }
}
```

注意此处的 `&` 符号，它表示 `li` 元素自身，上述的表达式相当于:

```
li:last-child {}
```

第二天

我们在第一天已经学习了很多的基础知识，而且学会了现代的前端开发流程（使用LiveReload节省了我们

很多的时间，使用Emmet编写HTML更是`键动如飞`）。同时，我们还一步步的开发了`Find an agent`页面的`Hero区域`。

但是这个区域并没有完全和设计图匹配，比如颜色，位置，字体等，我们今天来接着完成。

在开始之前，我们可以先来学习另外一些基础知识。

字体

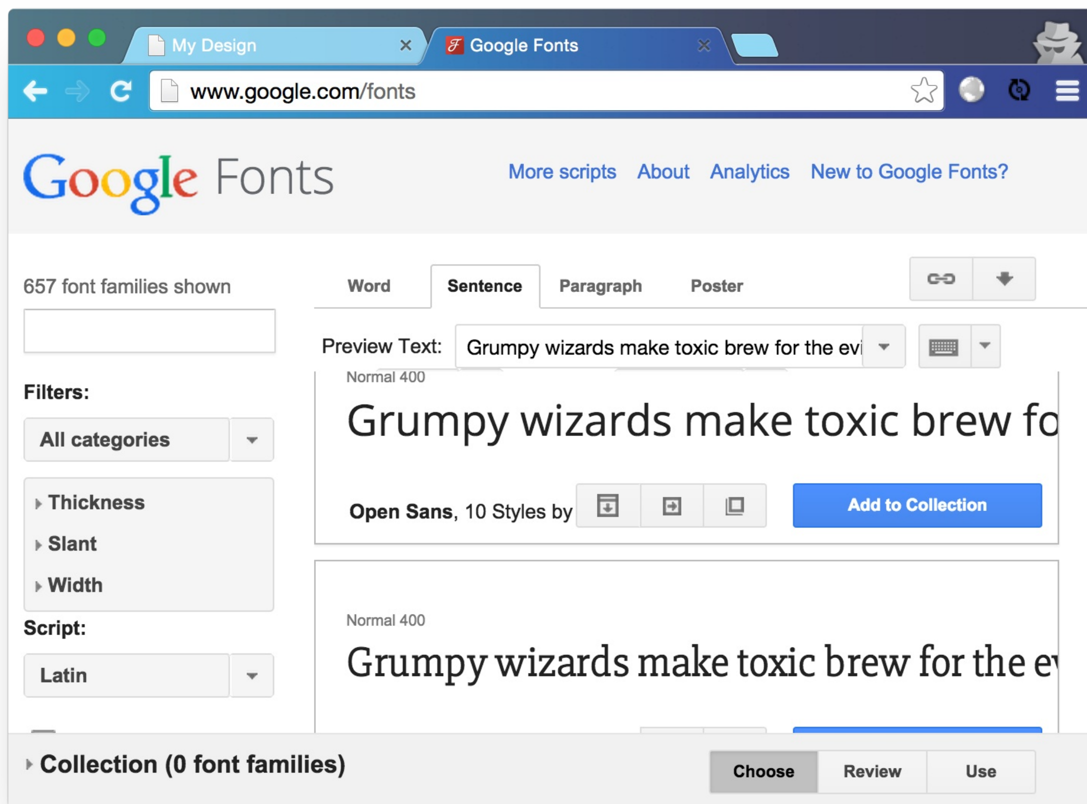
研究发现，Web设计，有95%都是与排版相关。换言之，不论你的设计是专注于哪个方面，`内容`都是最重要的。而作为展现内容的字体，则毫无疑问的非常重要。

浏览器默认的字体非常简单，在可读性上并没有问题，但是并不美观。字体在大的方面可以分为两类：衬线字体（Serif）和无衬线字体（Sans Serif）。所谓衬线，就是在文字上的一些修饰性的线，浏览器默认的字体就是衬线字体。

衬线字体是比较经典的字体，从古罗马文字始，就开始作为正式的印刷字体而存在。无衬线字体则非常简单，而且更加易读。两者的区别可以在下面这个图中看出：



根据名称来选择字体是非常困难的，我们可以使用Google Font API来简化这个过程。`Google Font API` http://www.google.com/fonts，我们只需要选好自己喜欢的字体，Google会生成一个link，然后我们在自己的页面中应用这个link就可以使用该字体了。



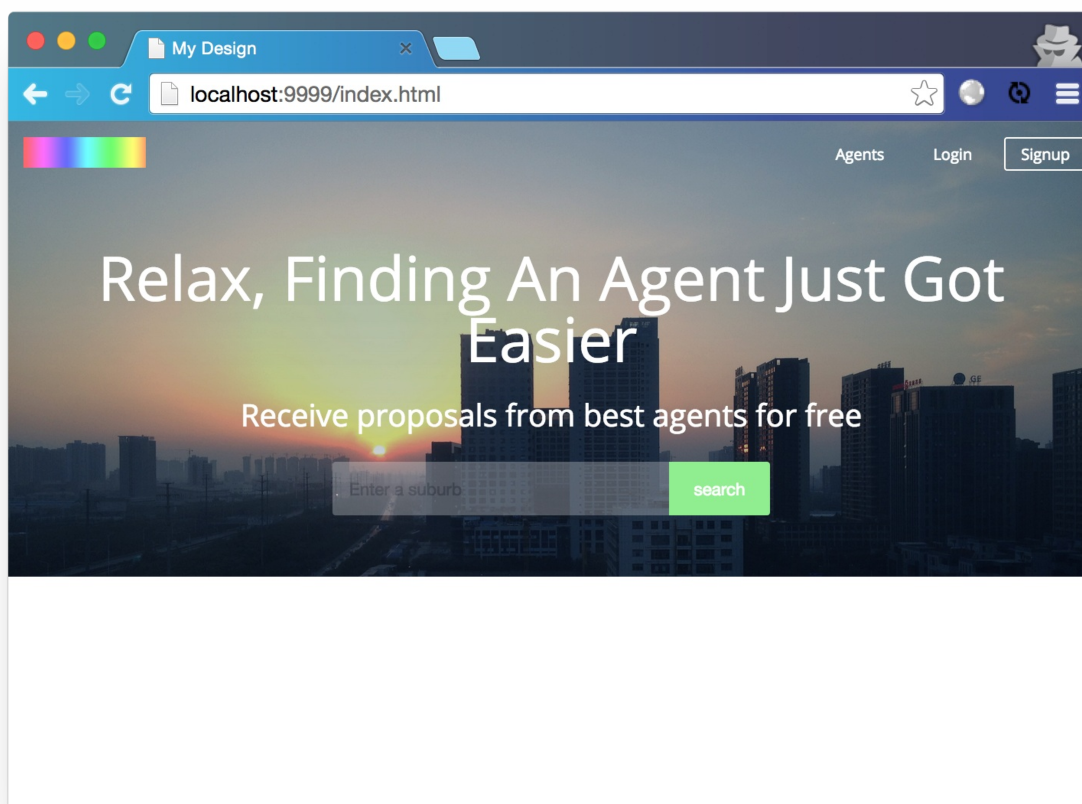
比如，我们选中了 `Open Sans` ，Google会生成一个对应的link：

```
<link href='http://fonts.googleapis.com/css?family=Open+Sans:300italic,400italic,600itali
```

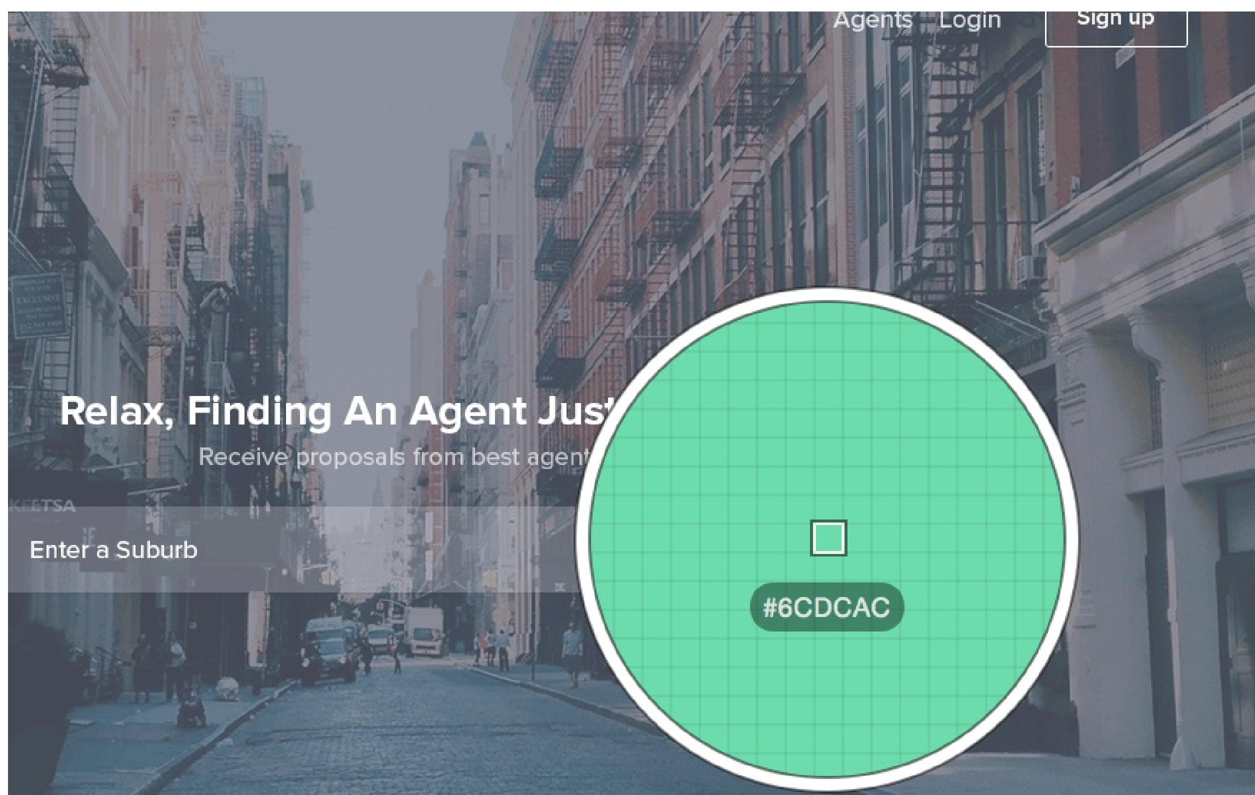
然后在 `scss` 文件中使用该字体

```
body {
    font-family: 'Open Sans', sans-serif;
    font-size: 62.5%;
    text-align: center;
}
```

应用了该字体之后，我们的界面会变得更加接近设计稿：



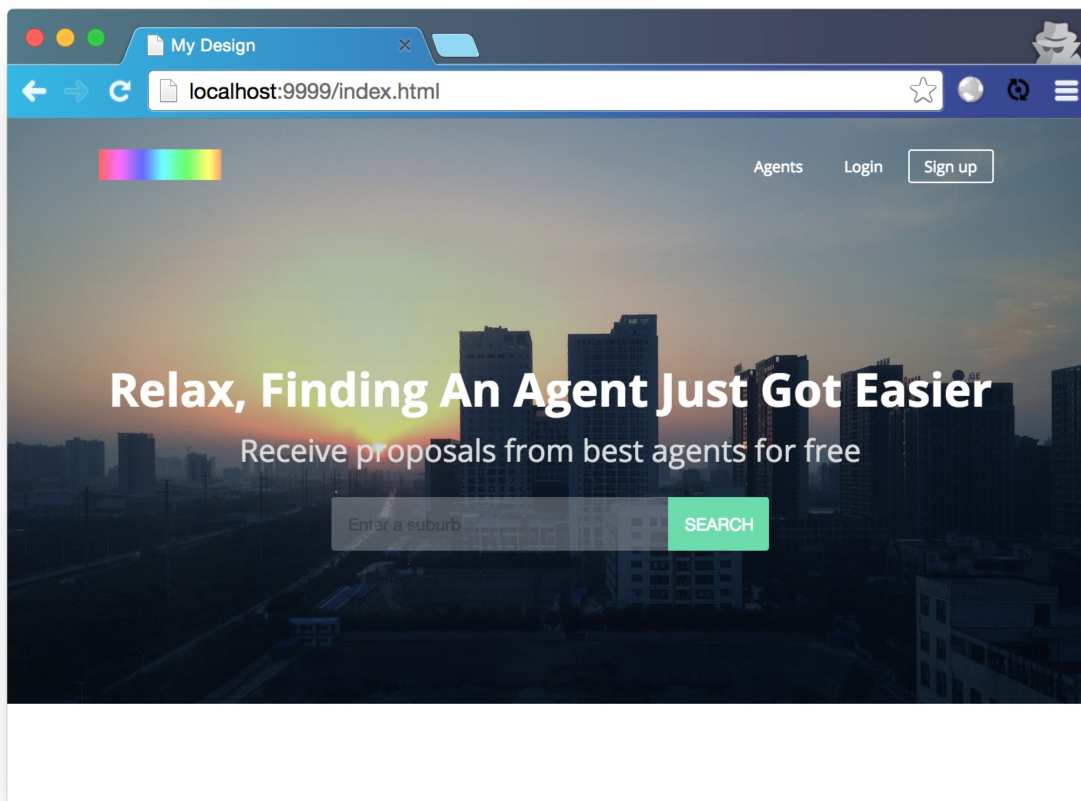
我们来接着第一天的成果，对页面元素进行一些细化。首先我们需要抽取设计稿上的颜色，有很多工具可以帮助我们做到这一点，比如在Mac OS X上有个工具 `SIP` ，使用它可以很方便的抽取页面元素的颜色，快捷键为 `Ctrl+Option+P` 。



抽取出颜色之后，将这些颜色定义为 `sass` 的常量：

```
$bg-color: #efefef;
$text-color: #4f4f4f;
$secondy-text-color: #D9DADA;
$highlight-text-color: white;
$button-color: #6cdcac;
```

然后将页面元素的位置做一些微调，会得到这样的效果：



一些需要注意的细节，比如我们通过 `text-transform: uppercase;` 来将搜索按钮上的文字变为大写；主标题的字体权重家中一号 `font-weight: bold;` ；副标题的颜色比主标题暗一些，颜色值为 `#d9dada` 。

到目前位置，页面已经相当接近设计稿了。我们可以暂时保持现状，然后开始后边部分的开发。按照顺序，我们下面来实现数字区域。

这个区域很清晰的可以用一个列表来实现，列表的长度为4，每个条目中都有两个标题，一个数字，一个说明：

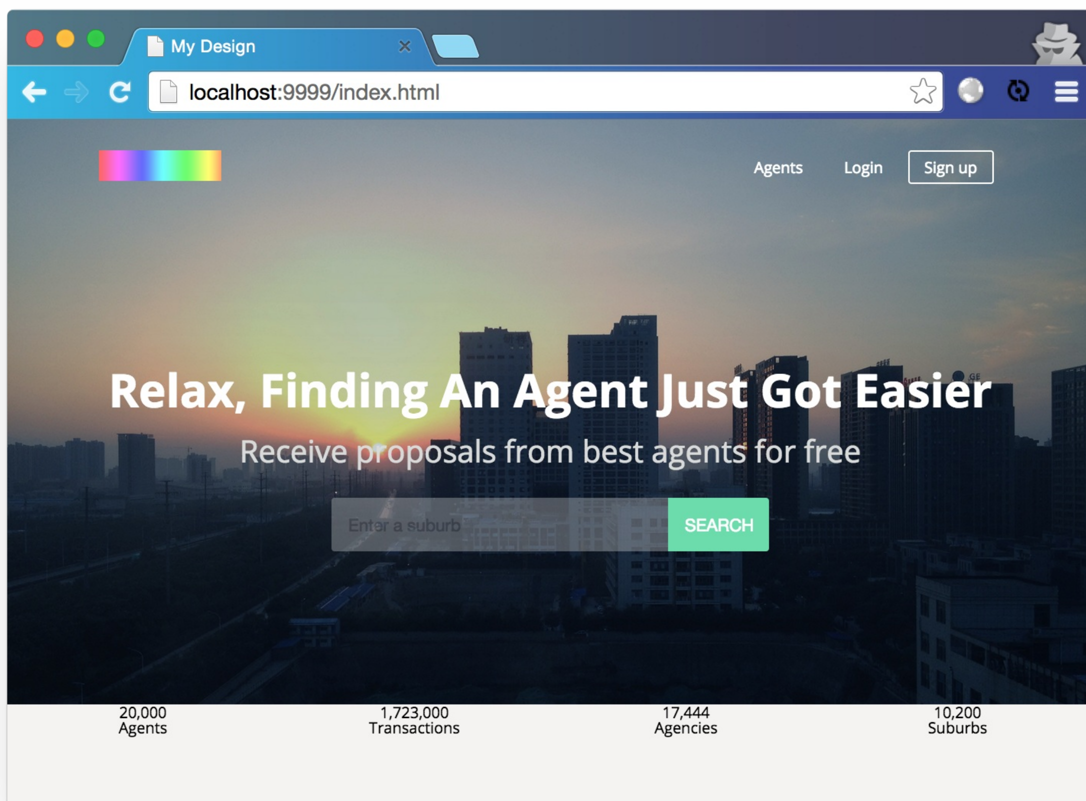
```
<section class="numbers">
    <ul>
        <li class="number">
            <h2>20,000</h2>
            <h3>Agents</h3>
        </li>
        <li class="number">
            <h2>1,723,000</h2>
            <h3>Transactions</h3>
        </li>
        <li class="number">
            <h2>17,444</h2>
            <h3>Agencies</h3>
        </li>
        <li class="number">
            <h2>10,200</h2>
            <h3>Suburbs</h3>
```

```
            </li>
        </ul>
    </section>
```

```
section.numbers>ul>li*4>h2+h3
```

首先，我们需要让这个列表横向排列，这个很容易，只需要将他们设置为浮动即可：

```
li {
    float: left;
    width: 25%;
}
```



而根据设计稿中可以看到，数字区域并没有占用100%的宽度，而是占用了80%左右。因此我们需要将 `li` 的父元素设置一个宽度，并使得其居中对齐：

```
.numbers {
    width: 80%;
    margin: 0 auto;
}
```

接下来，我们需要整个数字区域浮动起来，它有一半在Hero区域中，另一半则在 `How it works` 的区域。

```
.numbers {
    width: 80%;
    margin: 0 auto;

    ul {
        position: relative;
        top: -1em;
        background: $bg-color;

        li {
            float: left;
            width: 25%;
        }

        @include clearfix;
    }
}
```

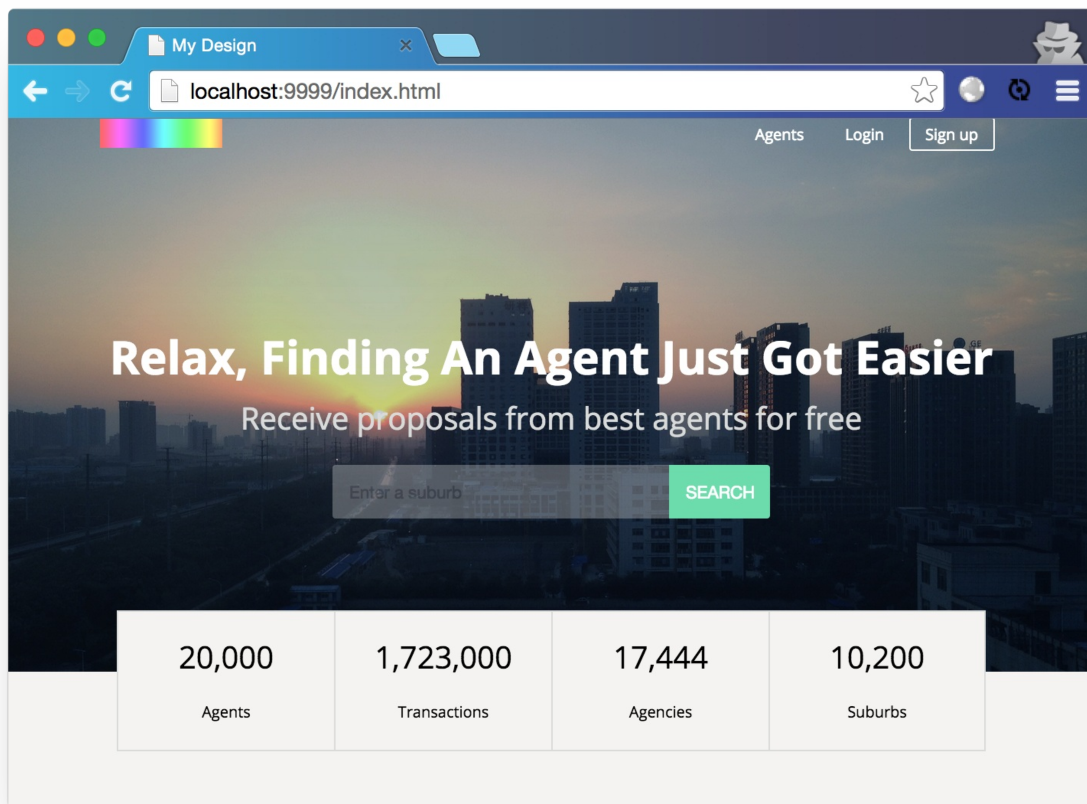
然后我们开始细化整个数字区域，调整字体，内间距，外间距等：

```
li {
    float: left;
    width: 25%;
    box-sizing: border-box;
    border-top: 1px solid $secondy-text-color;
    border-bottom: 1px solid $secondy-text-color;
    border-left: 1px solid $secondy-text-color;

    &:last-child {
        border-right: 1px solid $secondy-text-color;
    }

    h2 {
        font-size: 2em;
        padding: 1em 0;
    }

    h3 {
        padding: 0 0 2em 0;
    }
}
```



接下来我们就可以开始 `How it works` 部分的开发了，根据上面的经验，我们事实上可以很容易的将HTML写出来：

```
<section class="how-it-works">
    <h2>How it works</h2>

    <ul>
        <li>
            <div class="detail">
                <span class="number">1</span>
                <i class="icon-list"></i>
                <h3>Compare</h3>
                <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Earum tempor
            </div>
        </li>
        <li>
            <div class="detail">
                <span class="number">2</span>
                <i class="icon-pencil"></i>
                <h3>Receive Proposals</h3>
                <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Earum tempor
            </div>
        </li>
        <li>
            <div class="detail">
                <span class="number">3</span>
                <i class="icon-users"></i>
```

```
                <h3>Sell with the Best</h3>
                <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Earum tempor
            </div>
        </li>
    </ul>
</section>
```

这段HTML片段中有很多`Lorem`开头的随机文本，这些文本可以通过`Sublime`自动生成，你只需要在编辑器中输入`lorem`，然后按`TAB`键即可。这在进行Web设计时非常方便。

观察设计稿，我们发现了三个不同颜色的图标。我们当然可以将这三个图标从图片中截取出来，或者找到设计师，让他提供给我们。但是事实上还有一种选择，就是使用`字体图标`（fonticon）。

字体图标

字体图标，顾名思义，是一个字体，但是看起来又是一个图标。要使用字体图标，需要下载一个/一组字体文件，然后在CSS文件中使用该字体中的某些字符（通常这些字符都不会和正文所用的文字冲突，他们会选择Unicode中的一些保留码）。



目前已经有很多字体图标，其中有收费的，也有很多免费的。我们这里会使用免费的icomoon，同类型的还有fontawesome等。

在icomoon上选择一些自己需要的图标之后，选择下载。下载之后的包中包含了字体文件，和一个示例。首先需要将`fonts`目录拷贝到我们的工程中，其中会包含这样几个文件：

```
$ ls fonts
icomoon.eot  icomoon.svg  icomoon.ttf  icomoon.woff
```

然后将`style.css`拷贝到`sass`目录中，并重命名为`icomoon.scss`。最后，在`style.scss`文件中引入这个`icomoon`文件：

```
@import "icomoon";
```

你可能需要根据实际的位置修改`icomoon.scss`文件中相对`fonts`的路径：

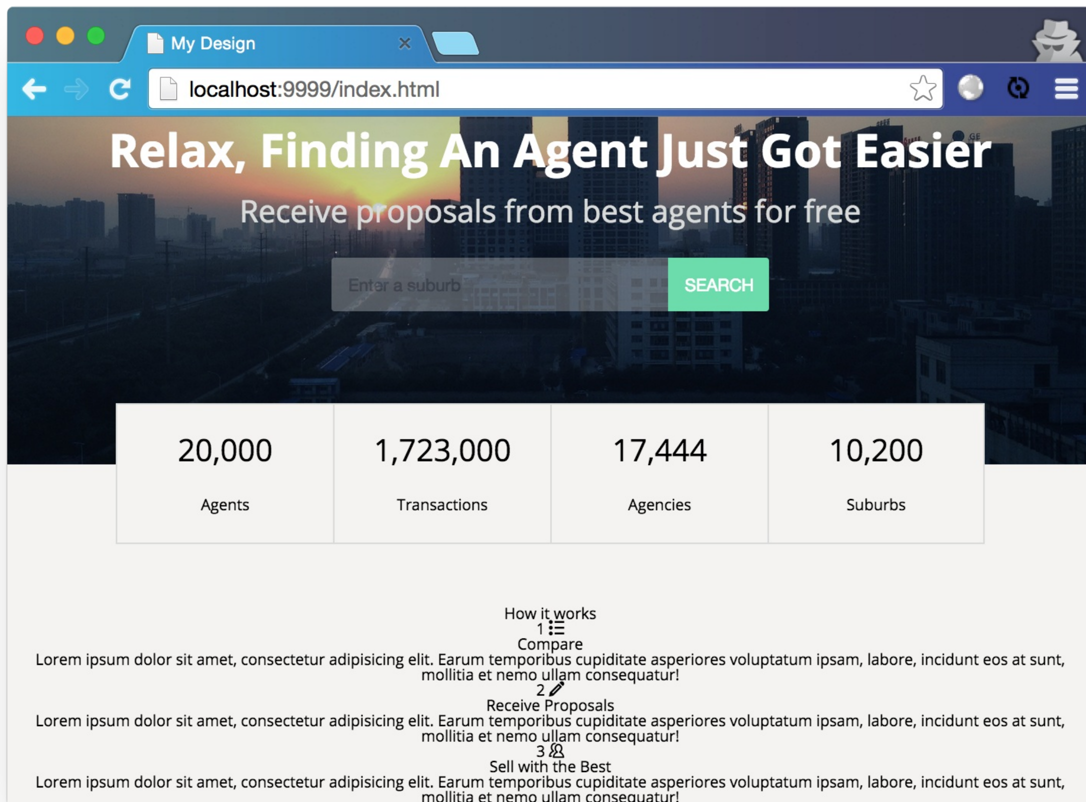
```
@font-face {
    font-family: 'icomoon';
    src:url('../fonts/icomoon.eot?200299');
    src:url('../fonts/icomoon.eot?#iefix200299') format('embedded-opentype'),
        url('../fonts/icomoon.woff?200299') format('woff'),
        url('../fonts/icomoon.ttf?200299') format('truetype'),
        url('../fonts/icomoon.svg?200299#icomoon') format('svg');
    font-weight: normal;
    font-style: normal;
}
```

如果你观察过我们之前的HTML片段，就会发现其中有一些`i`标签，这些标签的`class`都是以`icon-`开始的。你会在`icomoon.scss`中找到实际的定义：

```
.icon-zoom-out:before {
    content: "\e62b";
}
.icon-zoom-in:before {
    content: "\e62c";
}
```

此处的`content`实际就是一个unicode的编码，对应在字体文件中的一个图标。

这些图标展现出来正如图片一样：



字体图标的好处在于，它的尺寸比较小，而且可以适应各种屏幕尺寸，由于它本身是矢量的，无论缩放到何种比例，都不会发生模糊。 另外，由于它本身是字体，那么设置颜色又会变得非常容易，就像修改文本颜色一样。

对于每个条目，我们首先还是让其浮动起来，然后设置宽度为 `1/3` ：

```
li {
    float: left;
    width: 33.33%;
}
```

对于每一个数字，我们可以很容易的为其设置样式：

```
.number {
    font-size: 2em;
    color: $border-color;
    border: 1px solid $border-color;
    width: 1em;
    height: 1em;
    display: block;
    margin: 1em auto 2em auto;
    padding: .4em;
    @include border-radius(100%);
    background-color: $bg-color;
```

```
}
```

对于每个图标，它们都有不同的颜色：

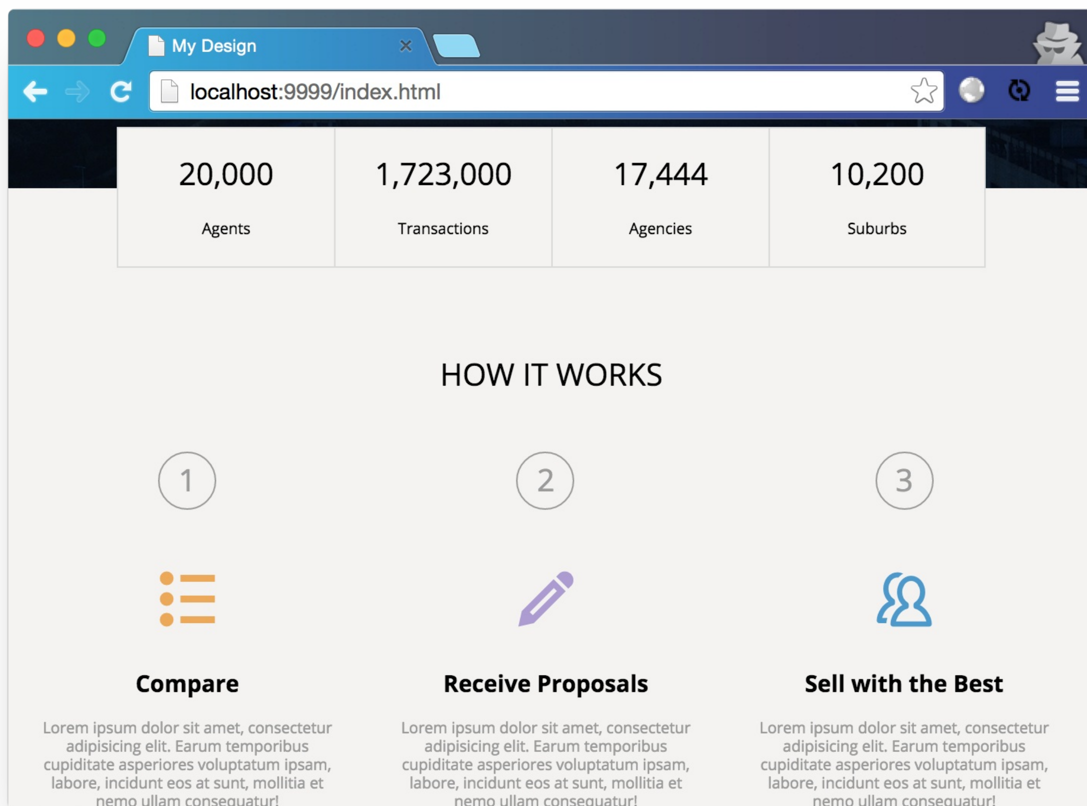
```
i {
    font-size: 3.6em;
}

.icon-list {
    color: #ECA95A;
}

.icon-pencil {
    color: #AC9CD2;
}

.icon-users {
    color: #4D99CB;
}
```

定义好这些之后，整体上的效果已经有了，数字，图标，描述等：



对照之前的设计稿，会发现很多细节的地方有所缺失，比如 `How it works` 下面的一个灰色指示条，三个数字之间的连线等，这些细节非常重要。我们有两种方式可以实现这些细节，第一种是修改HTML文档，添

加一些额外的元素，这种做法的缺点是为了展现而修改了内容。HTML文档本身仅仅应该包含内容，而不应该包含样式。因此我们更倾向于使用第二种方式：伪元素。

伪元素

伪元素是一个不存在在HTML代码中的元素，但是你又可以为其设置样式。这就给了我们一种方便的机制，在不影响HTML文档结构的情况下，加入一些`伪`元素，并为其设置样式，这样就可以很好的解决我们上面遇到的问题。

每一个HTML元素都可以有两个伪元素，这两个伪元素可以分别使用`:before`和`:after`来选中：

```
p::before {
    content: "";
    display: block;
    color: #6e6e6e;
}
```

比如对于`How it works`这个标题，我们可以使用一个伪元素，然后为其设置颜色，位置等，使其和设计稿一致：

```
h2 {
    font-size: 2em;
    text-transform: uppercase;
    padding: 1em 0;

    &:after {
        display: block;
        content: "";
        background-color: $border-color;
        width: 2em;
        height: .1em;
        margin: .4em auto;
    }
}
```

这样，我们的`How it works`下面就会多出一条粗线：

HOW IT WORKS

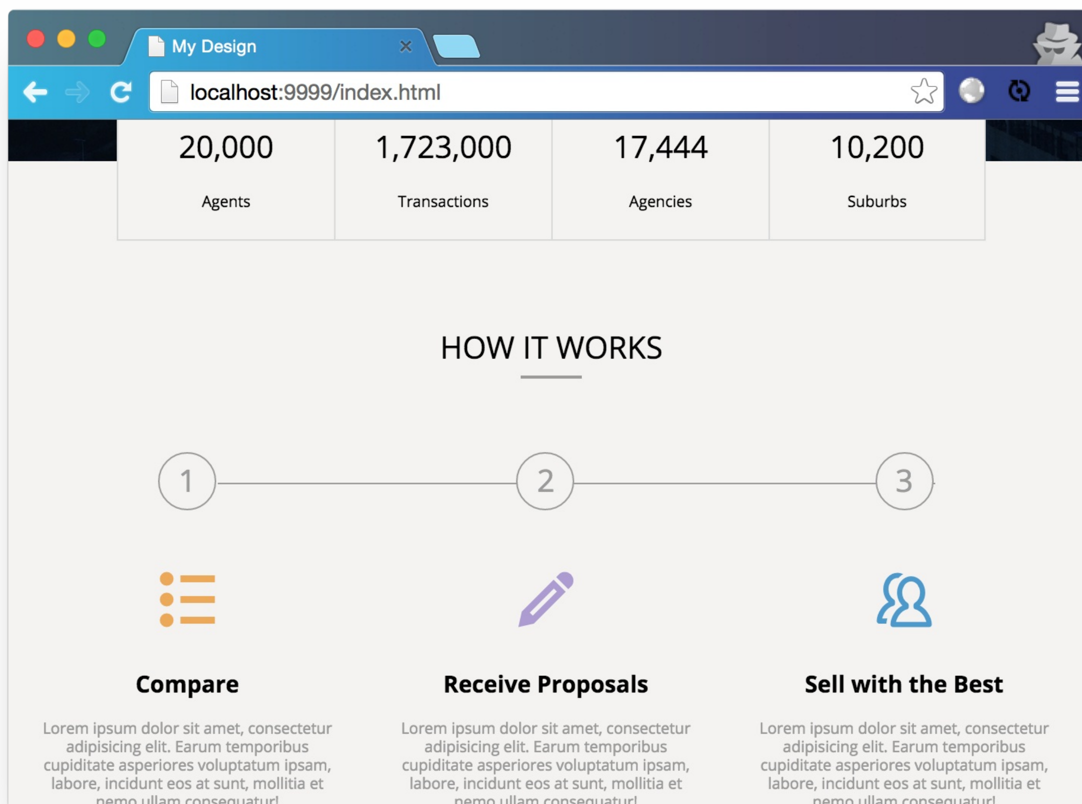
同样，我们为数字也加入伪元素：

```
li {
    &:before {
        display: block;
        content: "";
        background: $border-color;
        width: 100%;
        height: 1px;
```

```
        position: absolute;
        top: 4em;
        left: 50%;
        margin: 0 2em;
    }

    &:last-child:before {
        display: none;
    }
}
```

我们为每个li都添加了一个before伪元素，并为其设置了样式（一条灰色的直线），这样最后的那个li就会多出一条来，我们将其隐藏。最后的效果如下：



好了，我们第一天涉及的东西已经很多了，剩余的部分就交给读者自己练习了，使用上边这些原则和技巧，你应该很容易就可以实现页面的剩余部分。

发布你的设计

一个良好的工作流程是可以让你的读者或者用户尽快而且方便的看到你的成果。如果仅仅在本地开发，设计，你事实上很难发现自己的问题，从而难以得到真正的提升。而将自己的产出让别的开发人员或者设计师看到，并得到实时的反馈，则可以在很多方便帮助你改进。我们将使用 `Github` 的主页服务来发布自己的作品，当然如果你有亚马逊的云服务或者其他云服务提供商的主机，你也可以很容易的将作品发布在其中。

Github的`主页服务`

Github提供了Github Pages的服务来帮助你为自己的项目提供`主页`。目前，这种主页服务分为两种：`用户主页`和`项目主页`。其中`用户主页`已经称为广大开发者的标配，有很多的开发者已经将自己的博客迁移到了Github上，其中所用到的核心机制就是`Github Pages`。

这篇文章主要介绍如何使用`项目主页`。`项目主页`，顾名思义，就是你项目的主页，本来设计的初衷是为你的项目编写介绍文档，不过Github只提供对静态内容的托管。如果需要添加评论，可以使用disqus的服务，而和微博，flickr等集成都有现成的JavaScript片段，这里也不做详细讨论。

发布你的Web设计

Github提供的`项目主页服务`可以帮助你快速将设计发布，你所需要做的就是为项目创建一个名叫`gh-pages`的分支，然后将`HTML/CSS/JS`放在这个分支上即可。

假设你在github上的用户名为`wumai`(我一时间想不到好名字，在写这一页的时候看了看窗外，雾霾四合，不放就假设用户名为雾霾吧)，那么根据惯例，你的Github地址为`https://github.com/wumai`。这时候，假设你的项目（repo）的名称为`design-1`，则你的`项目主页地址`为`https://wumai.github.io/design-1`。

知道了你的`项目主页地址`，你就需要为这个页面添加内容了：

```
$ git clone git@github.com:abruzzi/design-boilerplate.git design-1
```

克隆了`design-boilerplate`之后，

```
$ cd design-1
$ git remote -v
```

你可以看到当前的项目是和`git@github.com:abruzzi/design-boilerplate.git`关联的，

```
origin    git@github.com:abruzzi/design-boilerplate.git (fetch)
origin    git@github.com:abruzzi/design-boilerplate.git (push)
```

你需要先和这个样板工程解除绑定：

```
$ git remote remove origin
```

然后你需要在Github上创建一个新的Repo，假设命名为`design-1`，这时候，将这个新创建的Repo作为你本地的remote：

```
$ git remote add -u origin git@github.com:wumai/design-1.git
```

与远程连接之后，我们可以开始实际的设计了，不过在这之前，需要先创建一个`gh-pages`分支：

```
$ git checkout -b gh-pages
```

这条命令会创建`gh-pages`分支，并切换到该分支，这样后续的修改都会在该分支进行，这也正是我们想要的。开发调试之后，就可以将这个分支push到Github：

```
$ git push -u origin gh-pages
```

好了，现在打开地址`http://wumai.github.io/design-1`，应该就可以看到你自己的设计了。



第二章：Twitters' New Face

今天我们将一起完成另外一个设计，`Twitter` 的新界面。和上一周的设计一样，它来源于 `dribble.com` 。



这个设计是一个非常典型的Web2.0应用模式，从内容上分析，它包括下面这样一些信息：

1. 用户本身的信息
2. 用户发布过的一些信息
3. 用户关注的其他用户发布的信息
4. 热门信息
5. 系统推荐的一些其他用户

这种模式在很多的设计中都可以用到，我们在实现的时候，顺便可以关注一下设计师是如何组织这些信息的。

第一天

在进入实际的开发之前，我们先来学习一些基本的知识，这些知识会帮助我们更好的实现页面，应用样式。

HTML5与CSS3

HTML5是下一代（新一代）HTML的标准。相较于HTML4，HTML5有了很多的修改，引入了一些新的标签，废弃了一些旧的标签，更新了一些标签的寓意等。它更关注HTML文档的语义化，使得HTML不仅仅可以被浏览器识别，更可能被爬虫或者其他应用使用等。

CSS3同样是对CSS2的扩展。它引入了众多的特性，比如自定义字体集，阴影，圆角，动画等等。在引入这些特性之前，Web开发者经常需要自己做一些`绕过`工作，比如实现阴影的时候会使用一张背景图片来实现，而圆角则需要为4个角都做一张小图片来拼凑起来。

HTML5的新标签

我们在这一小节主要讨论这样几个新的标签

1. NAV
2. HEADER
3. ASIDE
4. SECTION
5. ARTICLE
6. FOOTER

从标签的名称也可以看到HTML5在文档的语义化方面的改善。这些标签的引入，使得HTML文档更像一个文档。下面我们来分别介绍这6种标签。

NAV标签

`nav`标签的语义为：当前文档的导航。`nav`作为一个容器，包含了链接到其他文档或者当前文档的其他部分的链接。比如他可以被实现为当前站点的菜单栏的容器：

```
<nav>
    <ul>
        <li><a href="/product">Product</a></li>
        <li><a href="/contact">Contact</a></li>
        <li><a href="/detail">Detail</a></li>
    </ul>
</nav>
```

另外一个可能的场景是，在页脚上，通常会放置一些子站点的链接，这些链接也可以放置在`nav`中。比如`REA`是澳洲房产搜索的一个平台，它旗下有一系列的站点，比如商业地产，投资等，除了澳洲之外，它在意大利，香港等地也有站点。



International: France | Germany | Hong Kong | Italy | Luxembourg | New Zealand

Partners: news.com.au | Property for sale | Italian property partner

© REA Group Ltd (REA:ASX)

HEADER标签

`header` 标签的语义为：当前文档的头信息或者当前文档中section的头信息。比如上周中我们的页面上的 `Hero Section` 中的两个标题：

```
<header>
    <h1>Relax, Finding An Agent Just Got Easier</h1>
    <h2>Receive proposals from best agents for free</h2>
</header>
```

或者在 `section` 中也可以使用：

```
<section>
    <header>
        <h1>This is my title</h1>
        <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Eligendi officia min
    </header>
</section>
<section>
    <header>
        <h1>This is another</h1>
        <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Molestiae voluptatem
    </header>
</section>
```

虽然它是 `header` ，但是并不一定在物理上位于文档的头部，它更多的是一个逻辑概念。

ASIDE标签

`aside` 的语义为：对于那些与当前文档内容相关，但又不是主要内容的内容，可以放置在 `aside` 中。通常我们会将这种内容放置在 `侧栏` 中，但是应该注意的是，侧栏并不和 `aside` 完全对应。侧栏是一个视觉设计，而 `aside` 是一个逻辑概念。

`aside` 可以是相对于整个文档的，也可以是相对于某个seciton/article的：

```
<article>
    <h1>Dont design, redisign</h1>
```

```
    <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit.</p>
    <aside>
        <h1>Colour theory</h1>
        <dl>
            <dt></dt>
            <dt></dt>
        </dl>
    </aside>
</article>
```

在 `article` 中的 `aside` ，与 `article` 的相关性更高一些（相对于整个文档）。如果 `aside` 不在 `article` 中，那么它又应该是当前页面的次要内容。

```
<body>
    <header>
        <h1>Dont design, redisign</h1>
        <h2>Lorem ipsum dolor sit amet, consectetur adipisicing elit.</h2>
    </header>
    <aside>
        <h1>Web design</h1>
        <dl>
            <dt></dt>
            <dt></dt>
        </dl>
    </aside>
</body>
```

SECTION标签

`section` 可以表示一个文档，或者一组逻辑上相关的内容。但是它并不是一个通用的容器，如果仅仅是为了样式而需要引入一个容器，那么推荐使用 `div` ，而 `section` 更多的是和内容本身相关（而不是展现）。

reveal.js是一个很好的例子，它是一个用HTML编写幻灯片的工具，每张幻灯片都是相对独立的，它被包含在一个 `section` 中：

```
<section>
    <h2>UI for developers</h2>
    <pre><code data-item>
        ***&lt;没钱赚商店&gt;购物清单***
        名称：雪碧，数量：2瓶，单价：3.00(元)，小计：6.00(元)
        名称：可口可乐，数量：4瓶，单价：15.00(元)，小计：60.00(元)
        ----------------------
        挥泪赠送商品：
        名称：雪碧，数量：1瓶
        ----------------------
        总计：60.00(元)
        节省：3.00(元)
        **********************
    </code></pre>
</section>
```



ARTICLE标签

`article` 的语义为：它表示页面上的一块独立内容，本身应该是字包含的（一个足以表意的实体）。并不局限在一篇文章这个狭义的 `article` 上。比如一篇博客，一条评论，一篇文章/新闻，一条微博等。这个独立的内容可以被分发到别处，或者被作为RSS的条目来使用。

我们今天要实现的每一天 `twist` 都可以使一个 `article` 。




```
<article class="post">
    <img src="images/juntao.png" alt="" class="avator">
    <header>
        <span class="name">Juntao</span>
        <span class="account">@juntao</span>
        <span class="time">2m</span>
    </header>
```

```
        <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Doloribus veniam quas, d
        <footer>
        </footer>
    </article>
```

FOOTER标签

`footer` 与 `header` 可以对照着看，它也不仅仅是狭义的页脚，还可以被使用在 `section` 或者 `article` 中。在理解这些元素的时候，尽量放在文档/内容的角度来考虑，而不是从视觉上考虑。比如 `footer` 并不一定非要放在页面的最底部。

```
<footer>
    <nav>
        <a href="/product">Product</a>
        <a href="/contact">Contact</a>
        <a href="/detail">Detail</a>
    </nav>
    <h2>CopyRight@2014</h2>
</footer>
```

CSS3新特性

CSS3引入了众多的新特性，使得Web设计变得非常容易。在此之前，人们需要借助JavaScript技巧，精心设计的图片等来完成一些设计上的效果。目前CSS3的众多特性已经为所有的主流浏览器所支持，在你的设计中，你也可以放心的使用这些特性了。当然，异常总是存在的，比如奇葩的 `IE` 系列（事实上，我本人总是有意无意的在躲避 `IE` ）。

在介绍其他例子之前，我们先定义一个基准，假设我们有一个很简单的Box：

```
<section class="container">
    <section class="box">
        I'm a box
    </section>
</section>
```



圆角

圆角风格可以让一个带有边框的元素看起来没有那么锋利，事实上苹果公司的很多设计都带有一些圆角，比如窗口，桌面图标等。

在CSS3中，实现圆角非常简单，只需要定义一个圆角的半径即可：

```
.rounded {
    border-radius: 8px 8px 8px 8px;
}
```

这四个数字分别表示：左上，右上，右下，左下。当然，如果四个角的圆角半径一样的话，可以简写为：

```
.rounded {
    border-radius: 8px;
}
```

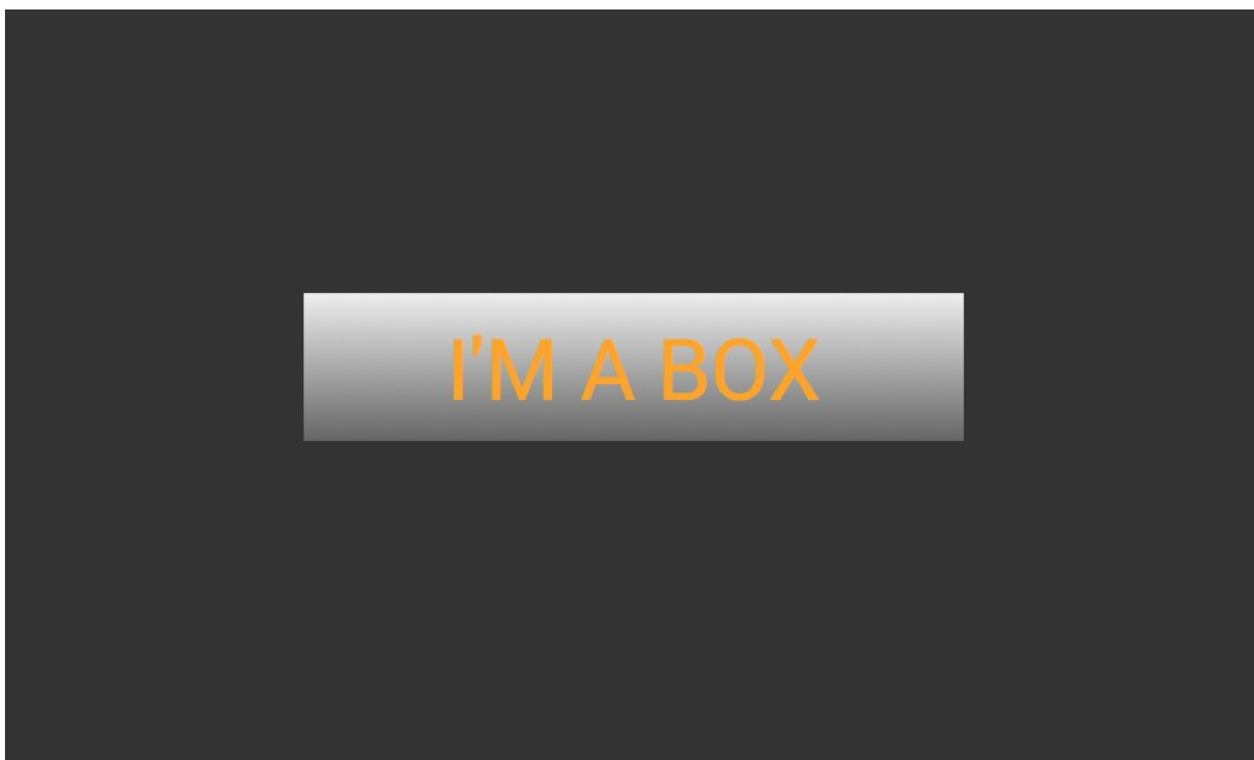


背景渐变

背景渐变可以以渐变的方式来修改一个元素的背景色，从而展现出立体的效果。背景渐变可以分为两类：`线性渐变`和`径向渐变`。线性渐变需要指定一个方向，然后一组颜色。

颜色值按照方向上的长度被分配，比如下面这个例子中我们使用了两个颜色：

```
.gradient {
    background: linear-gradient(to bottom, #eee, #666);
}
```



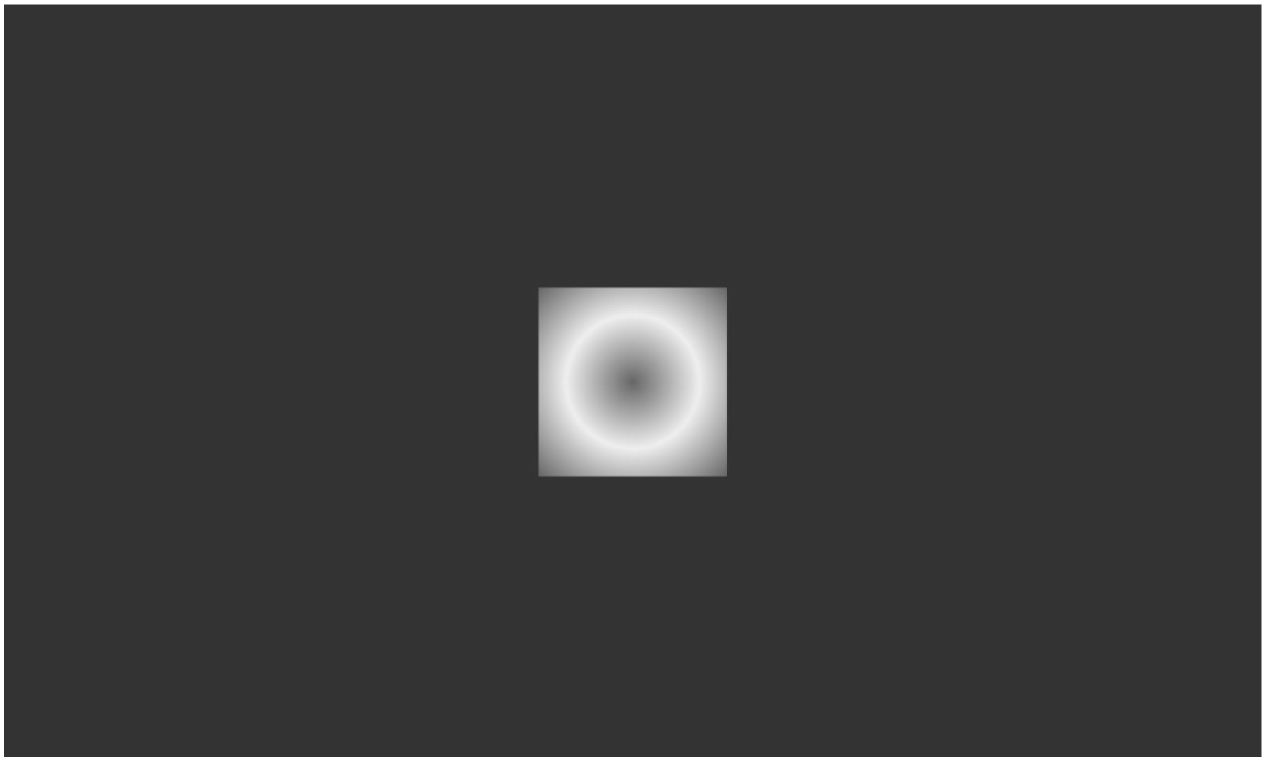
如果使用三种颜色：

```
.gradient {
    background: linear-gradient(to bottom, #eee, #666, #eee);
}
```



径向渐变默认的会从圆心开始，并逐渐按照径向来分配选中的一组颜色：

```
.gradient-rounded {
    background: radial-gradient(#666, #eee, #666);
}
```



盒阴影

盒阴影可以体现元素的立体感，在之前，要实现一个阴影，开发者需要绘制两个不同颜色的元素，然后叠加在一起。CSS3中的盒阴影可以大大简化这个过程。

定义一个盒阴影，需要指定这样一些参数：阴影在x轴上的偏移，阴影在y轴上的偏移，阴影的模糊半径，阴影的颜色。

```
.box-shadow {
    box-shadow: 5px 5px 0 orange;
}
```



上例中我们将模糊半径设置为了 `0`，从而得到了一个清晰的阴影，如果将模糊半径设置为 `5px`：



如果将偏移设置为负数，则可以得到一个在左上角的偏移：

```
.box-shadow {
    box-shadow: -5px -5px 5px orange;
}
```



盒阴影还有一个参数，可以实现内阴影：

```
.box-shadow {
    box-shadow: -5px -5px 5px orange inset;
}
```



文字阴影

和盒阴影相似，文字阴影用来为文字添加立体的效果。

```
.text-shadow {
    text-shadow: 2px 2px 1px gray;
}
```



过渡（**transition**）

过渡发生在元素在两个状态中切换时，它会`缓慢的`的将元素从一个状态变到另外一个状态。

比如我们上例中的`box`，假设其样式为：

```
.box {
    background: white;
    color: orange;
}
```

当`hover`时，其样式变成了

```
.box:hover {
    background: orange;
    color: white;
}
```

这时候从鼠标进入box到前景色/背景色互换，几乎没有延迟，这在某些情况下会变得比较烦人，特别是当鼠标在一个列表中通过时，会给人以眼花缭乱的感觉。`过渡`则可以让这个过程变得舒缓：

```
.box {
    background: white;
```

```
    color: orange;
    transition: all .5s ease-in-out;
}
```

这时候当鼠标进入box到实际颜色发生变化需要 `0.5` 秒，而 `ease-in-out` 是一个函数名，相应的还有 `ease-in` ， `ease-out` 等。他们的区别在时间很短的时候几乎觉察不到，大家可以在这里看到详细的解释。

变换

变换事实上为我们提供了基本的动画能力，比如我们在很多网站上都见到过这样的效果：当鼠标挪到一张图片的缩略图上时，图片会变大一些，挪开之后又会复原。当然，使用JavaScript也可以模拟这个动作，不过CSS3中已经自带了这样的特性。

在CSS3中使用变换非常简单，比如刚才提到的这个效果：

```
.element:hover {
    transform: scale(1.2);
}
```

当鼠标挪进该元素时，元素会变为之前的 `1.2` 倍大。另一个常见的变换效果是旋转：

```
.element:hover {
    transform: rotate(180deg);
}
```

当鼠标挪进该元素时，元素会旋转180度。我们最后再来看一个移动的变换：

```
.element:hover {
    transform: translate(200px, 100px);
}
```

当鼠标挪进该元素时，元素会向右下方移动（x轴方向200像素，y轴方向100像素）。CSS3事实上支持非常多的变换函数，这里就不一一列举了，读者可以参考这个详细的函数列表。

Marking up

好了，我们已经学习了足够多的理论知识了，下面就开始实际的页面开发吧，相信你已经摩拳擦掌，跃跃欲试了吧？

我们先来尝试编写整个页面的HTML结构，我们可以使用之前学习的HTML5的新标签，但是也不用彻底丢弃HTML4的标签：

```
<header>
    <nav>
        <ul>
            <li><a href="#" class="compose"></a></li>
```

```
                <li><a href="#" class="alert"></a></li>
                <li><a href="#" class="email"></a></li>
                <li><a href="#" class="tag"></a></li>
            </ul>
        </nav>
    </header>
    <aside>
        <div class="profile"></div>
        <div class="gallery"></div>
        <div class="activities"></div>
    </aside>
    <section class="content">
        <nav class="info">
            <ul>
                <li class="number">200</li>
                <li class="number">120</li>
                <li class="number">180</li>
                <li class="number">200</li>
            </ul>
        </nav>
        <div class="articles">
            <article></article>
            <article></article>
            <article></article>
        </div>
    </section>
    <aside>
        <div class="trends"></div>
        <div class="to-follow"></div>
    </aside>
    <footer>
        <div class="copyright"></div>
    </footer>
```

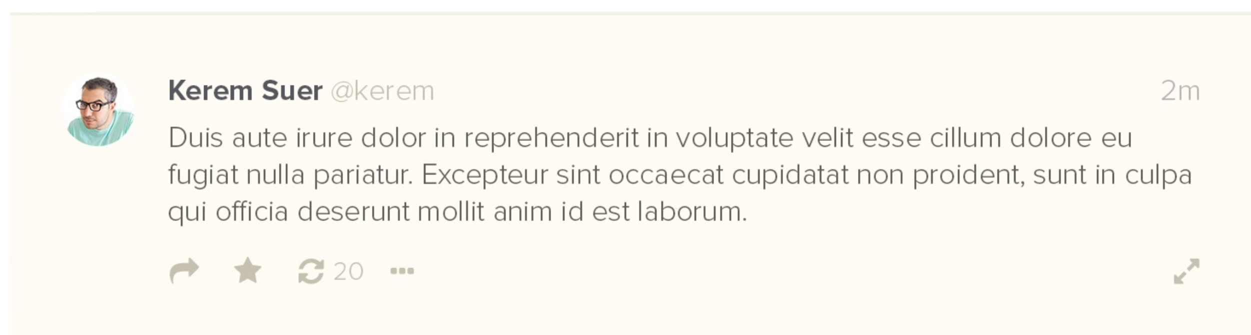
我们将页面分为5个主要的区域：头部，左边栏，内容区，右边栏，页脚。当然这只是一个初步的划分，如果我们发现有需要重新调整的地方，随时可以回来进行修改。

第二天

有了大的划分之后，我们就可以进行实际的开发了。我们先选择页面上的 `Tweet` 条目进行实现，这部分非常有代表性，很多其他Web设计都包含了这样的模式，因此学会这个可以帮助我们快速的实现页面的其他部分。

Tweet条目的实现

一个Tweet在设计上是这样的：



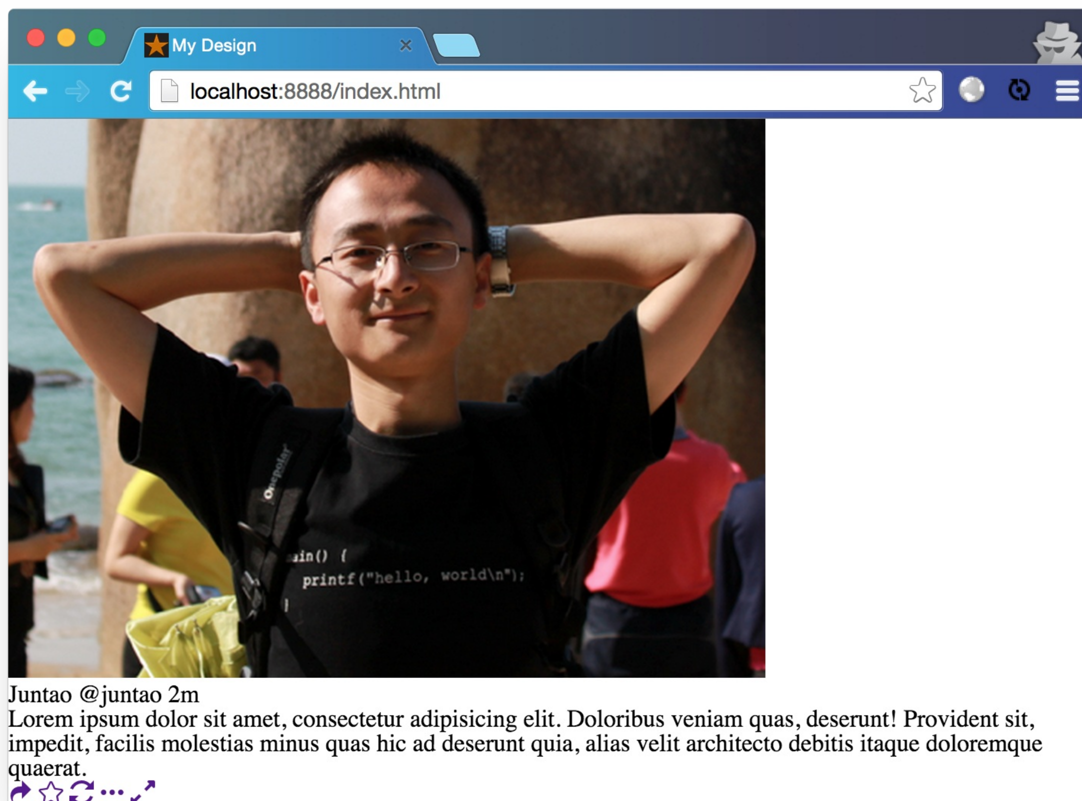
对于每一条Tweet，根据HTML5对 `article` 的描述，我们其实可以很自然的将Tweet实现为 `article` ，所以我们可以这样定义一个Tweet：

```
<article class="post">
    <img src="images/juntao.png" alt="" class="avator">
    <section class="content">
        <header>
            <span class="name">Juntao</span>
            <span class="account">@juntao</span>
            <span class="time">2m</span>
        </header>
        <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Doloribus veniam qua
        <footer>
            <nav>
                <a href=""><i class="icon-redo"></i></a>
                <a href=""><i class="icon-star"></i></a>
                <a href=""><i class="icon-loop"></i></a>
                <a href=""><i class="icon-ellipsis"></i></a>
                <a href=""><i class="icon-maximize"></i></a>
            </nav>
        </footer>
    </section>
</article>
```

每个 `Tweet` 中，首先需要一张作者的头像，然后是用户昵称，用户名，时间，正文，在正文下面有一些link，包括转发，赞等。这种布局方式非常常见，你可以仔细看一下上一周的设计，同样会发现这种模式：左边是一个独立的图片，右边是另外一大块的信息，右边的信息都严格的左对齐。

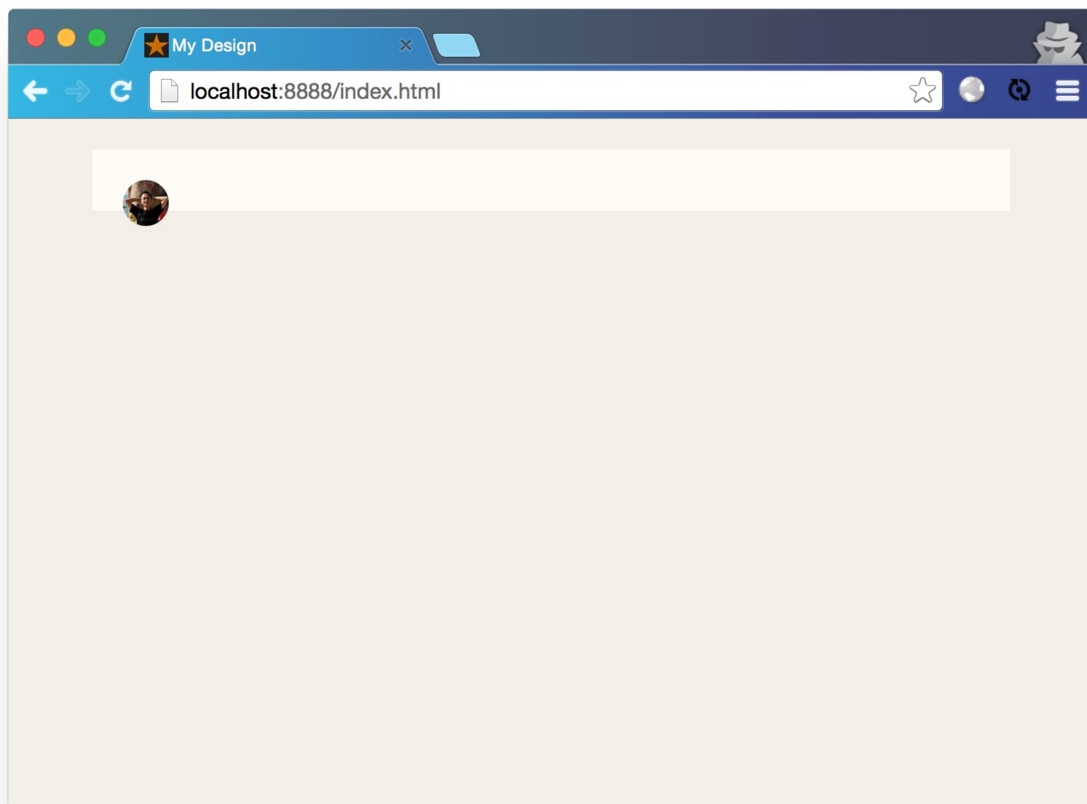
通常要实现这种布局，我们需要先为左右两部分设置一个相同的父元素，比如此处的 `post` 。然后将父元素的 `position` 设置为 `relative` ，将左边的元素的 `position` 设置为 `absolute` ，这样左边的元素就可以相对于父元素绝对定位了。最后，为右边的元素设置一个 `margin-left` ，空出一定的距离给左边的元素。

在未应用样式之前：



我们先来为头像加上样式，首先需要限定头像的大小（30x30），然后将border-raduis设置为100%，这样可以得到一个完美的圆。最后，将其对于父元素的偏移设置好：

```
.avator {
    width: 3em;
    height: 3em;
    @include border-radius(100%);
    position: absolute;
    top: 2em;
    left: 2em;
}
```



然后我们需要将用户昵称，用户名等浮动起来，并且将底部的那些可以点击的链接也浮动起来：

```
.post {
    position: relative;
    text-align: left;
    padding: 2em;
    background: $post-bg-color;
    border-bottom: 1px solid $body-color;

    .content {
        margin-left: 4em;

        span {
            margin-right: 1em;

            &.name {
                font-weight: bold;
            }

            &.account {
                font-weight: lighter;
                color: $gray-text-color;
            }

            &.time {
                color: $gray-text-color;
                float: right;
```

```
                margin-right: 0;
            }
        }
    }

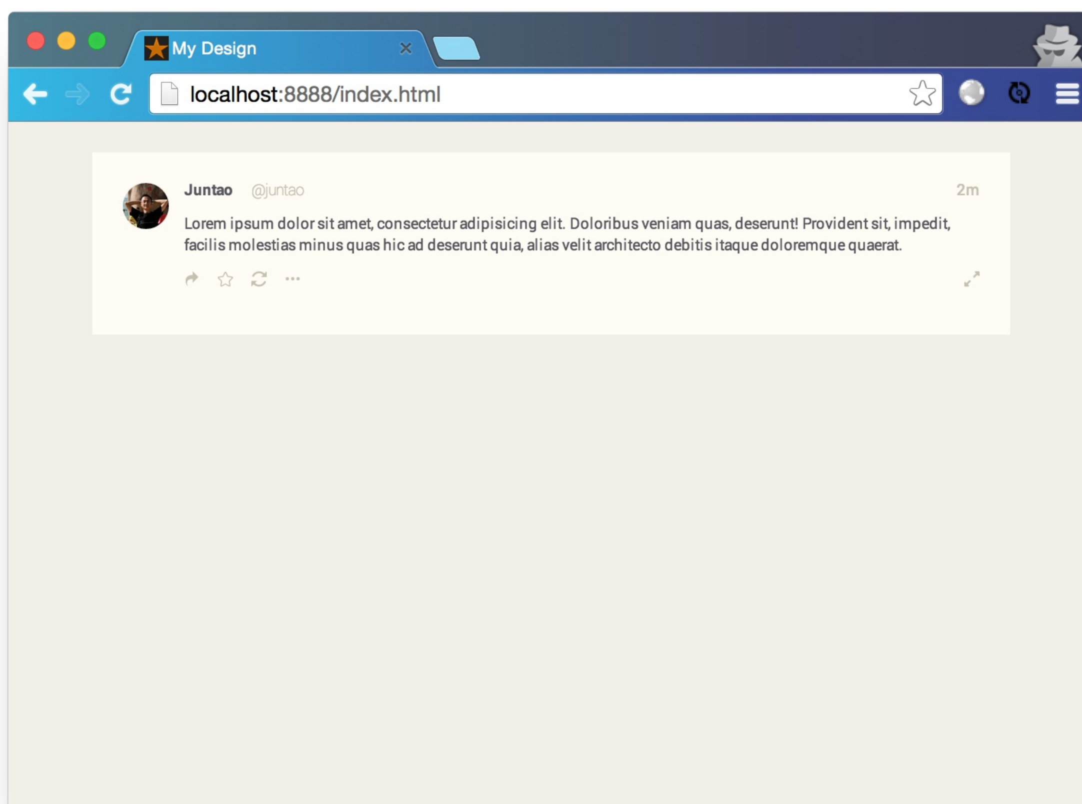
    header {
        margin-bottom: 1em;
    }

    section {
        margin-bottom: 1em;
        p {
            line-height: 1.4;
            font-weight: normal;
        }
        img {
            width: 100%;
            margin-bottom: 1em;
        }
    }

    footer {
        a {
            text-decoration: none;
            color: $gray-text-color;
            margin-right: 1em;
            &:last-child {
                float: right;
                margin-right: 0;
            }

            &:hover {
                color: $text-color;
            }
        }
    }
}
```

这样就可以得到一个非常接近设计稿的`Tweet`了：



CSS3颜色

在CSS2中，颜色可以通过三种方式来表示

1. 预定义颜色名，如 `red` ，`orange`
2. 十六进制，如 `#004c97`
3. RGB表示法，如 `rgb(255, 122, 0)`

颜色值有红，绿，蓝混合表示，每种颜色的取值都在区间0-255内，比如 `#004c97` 表示，红色值为 `0(0x00)` ，绿色值为 `76(0x4c)` ，蓝色值为 `151(0x97)` 。

在CSS3中又引入了三种新的颜色表示法，分别为rgba，hsl和hsla。rgba为rgb添加了alpha通道，即透明通道。

比如：

```
background-color: rgba(200, 120, 80, 0.2);
```

最后一个数字的取值范围在区间0-1之间，如果该值为0，则表示完全透明，值为1则表示完全不透明。

CSS3引入的HSL色彩空间是一个非常有意思的表示法。我们通常描述颜色的时候，不会直接转换成RGB的值，更多的时候我们会说：我想要这个按钮是亮一点的蓝色，或者当我们看到一种不太红的红色，我们直

观的想法是让它更偏向红色一些。但是这些描述无法通过RGB表示法表达出来，而HSL则可以做到这一点。

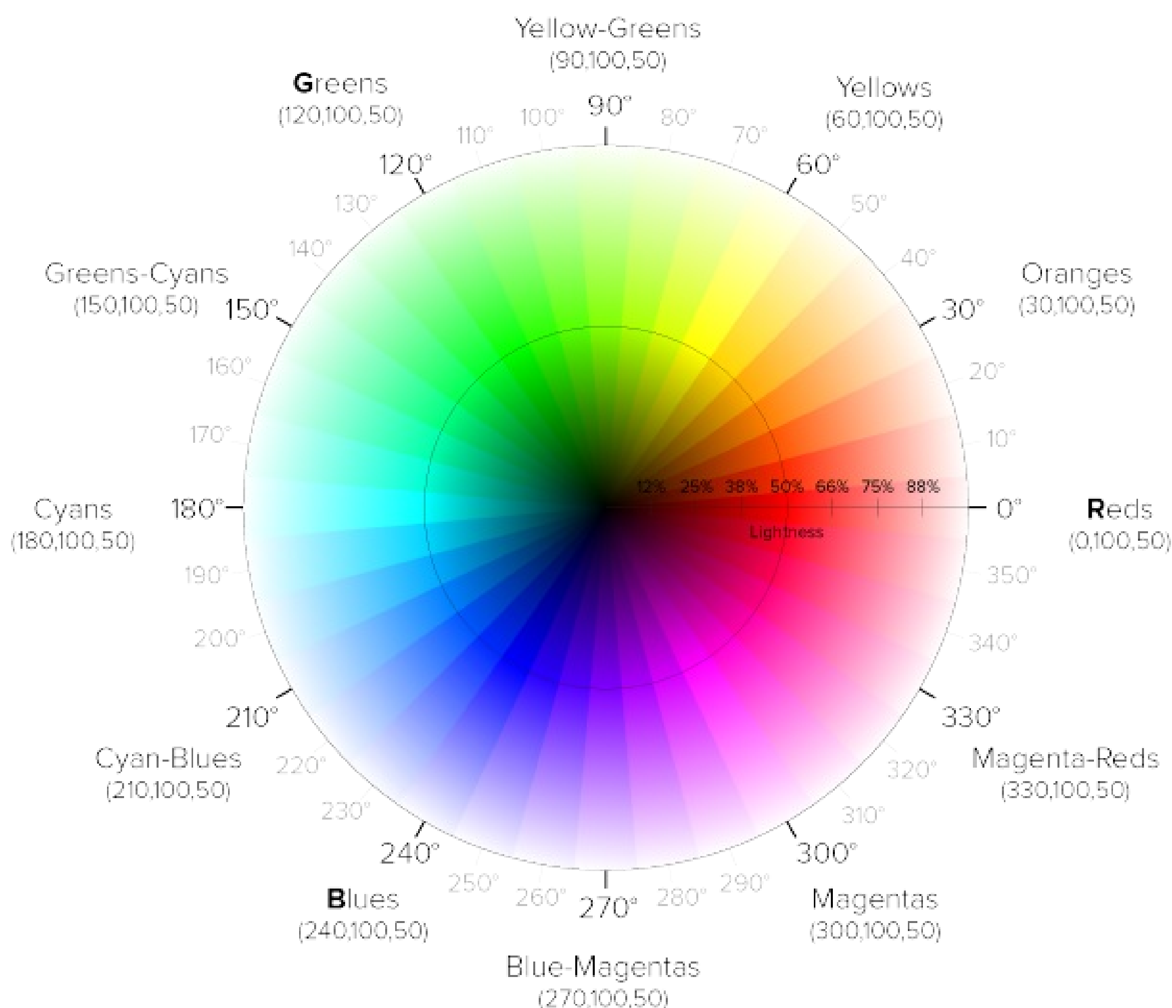
HSL分别表示Hue（色相），Saturation（饱和度），Lightness（亮度）。色相就是我们常说的红色，蓝色等；饱和度是指，在色相中参入的白色的程度，饱和度越高，色彩就越锐利，越接近色彩本身的值；亮度是在色相中参入黑色的程度，亮度越高越亮，越低越暗。我们可以通过两张图来描述清楚HSL色彩空间：



从上图可以看出HSL色彩的分布情况。使用HSL来表示颜色需要三个参数：

```
background-color: hsl(30, 100%, 50%);
```

上面的颜色定义指：色相为30，饱和度为100%，亮度为50%的颜色，色相在色盘上的位置如下图所示：



如果已知一个颜色的定义是 `hsl(200, 80%, 40%)`，比如用户反馈说这个色彩太暗，我们可以很容易的调整 `40%` 为 `45%` 或者 `50%` ；或者用户抱怨色彩不够深，不够蓝，我们事实上又可以调整色相后者饱和度。

响应式设计

随着不同尺寸的移动设备的出现，开发者需要完成一种设计，这种设计可以在不同的设备上都正常工作。

流式布局（完全按照百分比来定位，去除代码中关于尺寸的硬编码）可以在不同的较大的屏幕上很好的工作。在移动设备上，虽然流失布局也很好的显示，但是由于手机屏幕本身的限制，一个很小的百分比的元素并不真的可用。比如在桌上显示器上，20%是一个足够大的可视区域，但是在手机上可能就完全无法识别了。

因此，响应式设计应用而生，简而言之，响应式设计为每个不同尺寸的设备分别作了设计。比如在桌面上，每行可以显示4张图片，而在iPad上，每行显示两张，在手机上，每行只显示一张图片。这样可以保证对于不同尺寸的设备都提供很高的可用性。

Media Query

在CSS中，你可以知道屏幕的尺寸，并且根据尺寸的不同来应用不同的CSS文件，这个机制就是`media query`。比如我们可以在一个`link`标签中写这样的代码：

```
<link rel="stylesheet" media="screen and (max-width: 480px)" href="numbers-small.css" />
```

这个CSS会在最大视口（viewport）为480px的环境中生效。同样，我们可以在CSS代码中使用这样的表达式：

```
.information {
    background-color: red;
}

@media screen and (min-width: 1024px) {
    .information {
        background-color: yellow;
    }
}
```

所有的响应式设计都依赖于这个机制，一旦我们可以感知屏幕的尺寸，就可以为其应用不同的样式：将某些元素的宽度增大，隐藏某些元素等等。接下来我们来看一个实例，并从中学习如何在我们的页面中使用响应式设计。

一个实例

比如`Twitter new face`中，我们可以看到一些数字，我们可以将它改造成响应式的。

```
<section class="numbers">
    <ul>
        <li class="item">
            <h4>Tweets</h4>
            <p>200</p>
        </li>
        <li class="item">
            <h4>Photo/Videos</h4>
            <p>250</p>
        </li>
        <li class="item">
            <h4>Following</h4>
```

```
            <p>55</p>
        </li>
        <li class="item">
            <h4>Followers</h4>
            <p>180</p>
        </li>
    </ul>
</section>
```

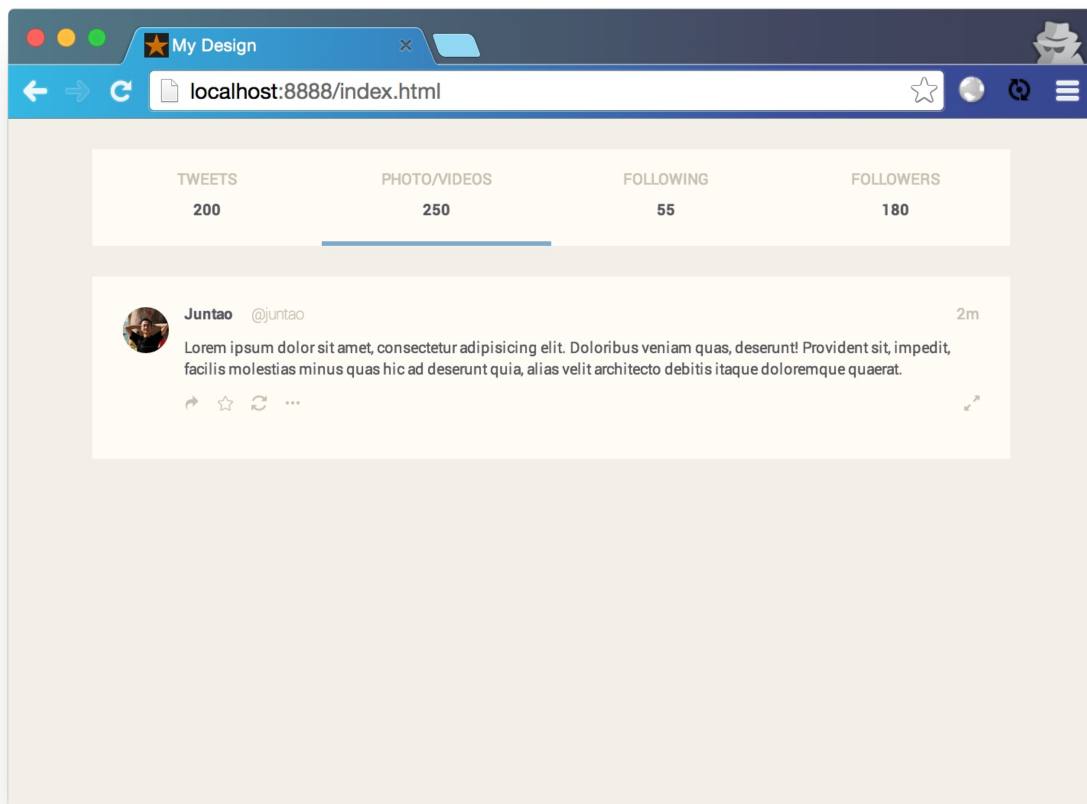
我们在桌面浏览器中，需要将这4个数字展现在同一行上：

```
.numbers {
    width: 60em;
    margin: 2em auto;
    background: $post-bg-color;
    li {
        float: left;
        width: 25%;
        box-sizing: border-box;
        padding: 1.5em 3em;
        border-bottom: 3px solid transparent;
        transition: all .3s ease-in;

        h4 {
            text-transform: uppercase;
            color: $gray-text-color;
        }

        p {
            margin-top: 1em;
            font-weight: bold;
        }

        &:hover {
            border-bottom: 3px solid $twitter-color;
        }
    }
    @include clearfix;
}
```



Sass在3.2之后，提供了非常方便的机制，我们可以根据不同的尺寸来应用不同的样式。首先定义一个 `mixin` ，这个 `mixin` 接受三个可能的参数，分别表示手机，平板和桌面浏览器。

```
@mixin respond-to($media) {
  @if $media == handhelds {
    @media only screen and (max-width: $break-small) { @content; }
  }
  @else if $media == medium-screens {
    @media only screen and (min-width: $break-small + 1) and (max-width: $break-large - 1
  }
  @else if $media == wide-screens {
    @media only screen and (min-width: $break-large) { @content; }
  }
}
```

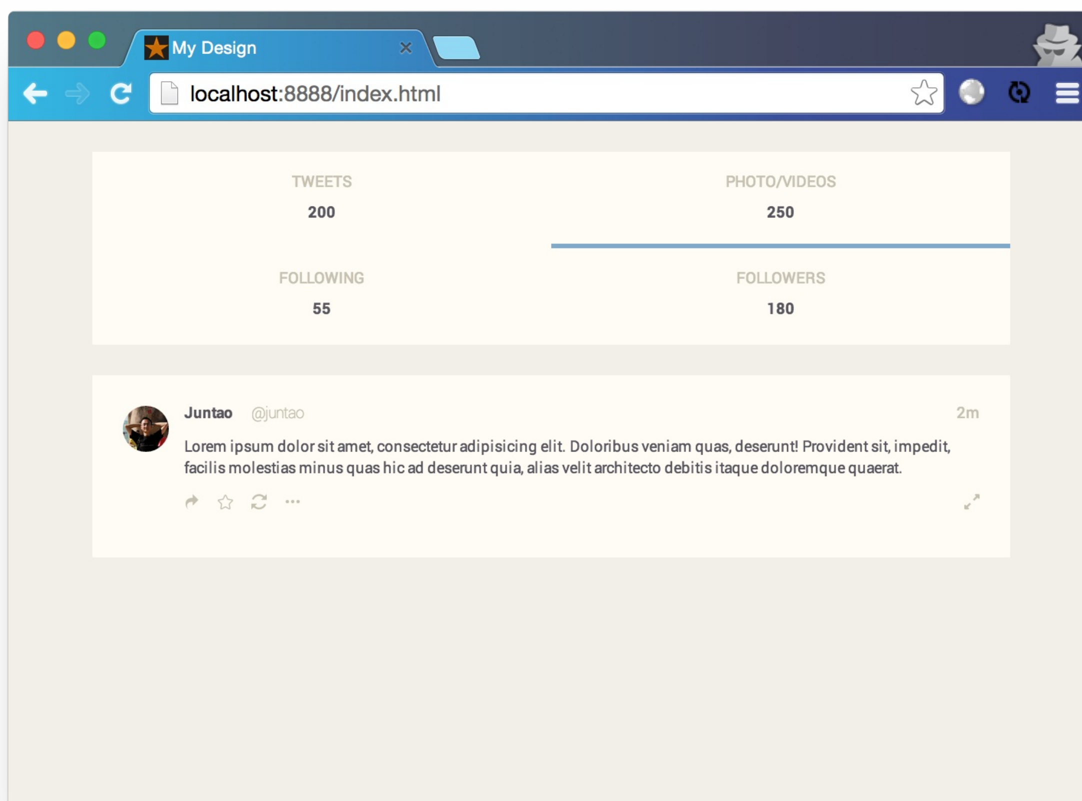
而我们需要先定义两个break-point：

```
$break-small: 320px;
$break-large: 1024px;
```

在使用的时候，只需要指定：

```
li {
    float: left;
    width: 25%;

    @include respond-to(medium-screens) {
        width: 50%;
    }
    //...
}
```



对于手机屏幕：

```
li {
    float: left;
    width: 25%;

    @include respond-to(handhelds) {
        float: none;
        width: 100%;
    }
    //...
}
```


iOS Simulator - iPhone 4s - iPhone 4s / iOS 8....
Carrier
8:48 PM
localhost
TWEETS
200
PHOTO/VIDEOS
250
FOLLOWING
55
FOLLOWERS
180
Juntao
@juntao
2m
Lorem ipsum dolor sit amet, consectetur adipisicing elit. Doloribus veniam quas, deserunt! Provident sit, impedit, facilis molestias minus quas hic ad deserunt quia, alias velit architecto debitis itaque doloremque

第3周：设计你自己的页面

经过了前两周的练习，你已经具备了实现一个既有设计的能力。你应该可以比较轻松的将一个设计稿`翻译`成HTML+CSS的实现了。但是这还不够，我们希望做的是自己从头实现一个自己的设计。

第一天

我们在第一天需要学习一些设计的原则，设计并不需要`天赋异禀`，它和其他技能一样，需要长期的锻炼。不过，已经有很多优秀的设计师总结了一些基本的模式，我们只需要沿着这些模式进行自己的设计，虽然未必能做到技惊四座，但是至少不会难看。

好的设计

我们在讨论`好`的设计时，它应该至少具有下列的特性：

1. 可用
2. 易用
3. 简单，直观
4. 美观

事实上，最优秀的设计就是让用户感觉不到设计本身的存在，即`人机交互的界面`融入到了日常的习惯中了。其实在很多时候，只有某个设计十分糟糕时用户才能体会到设计的存在：你找遍了整个页面却没有发现`重新发送邮件`的按钮；进入了一个深的层次后发现无法回退；一个鼠标移在上边不会变成`手`的链接等等。

我们这里来学习几条简单而容易使用的原则：对齐，对比，重复和相关性。

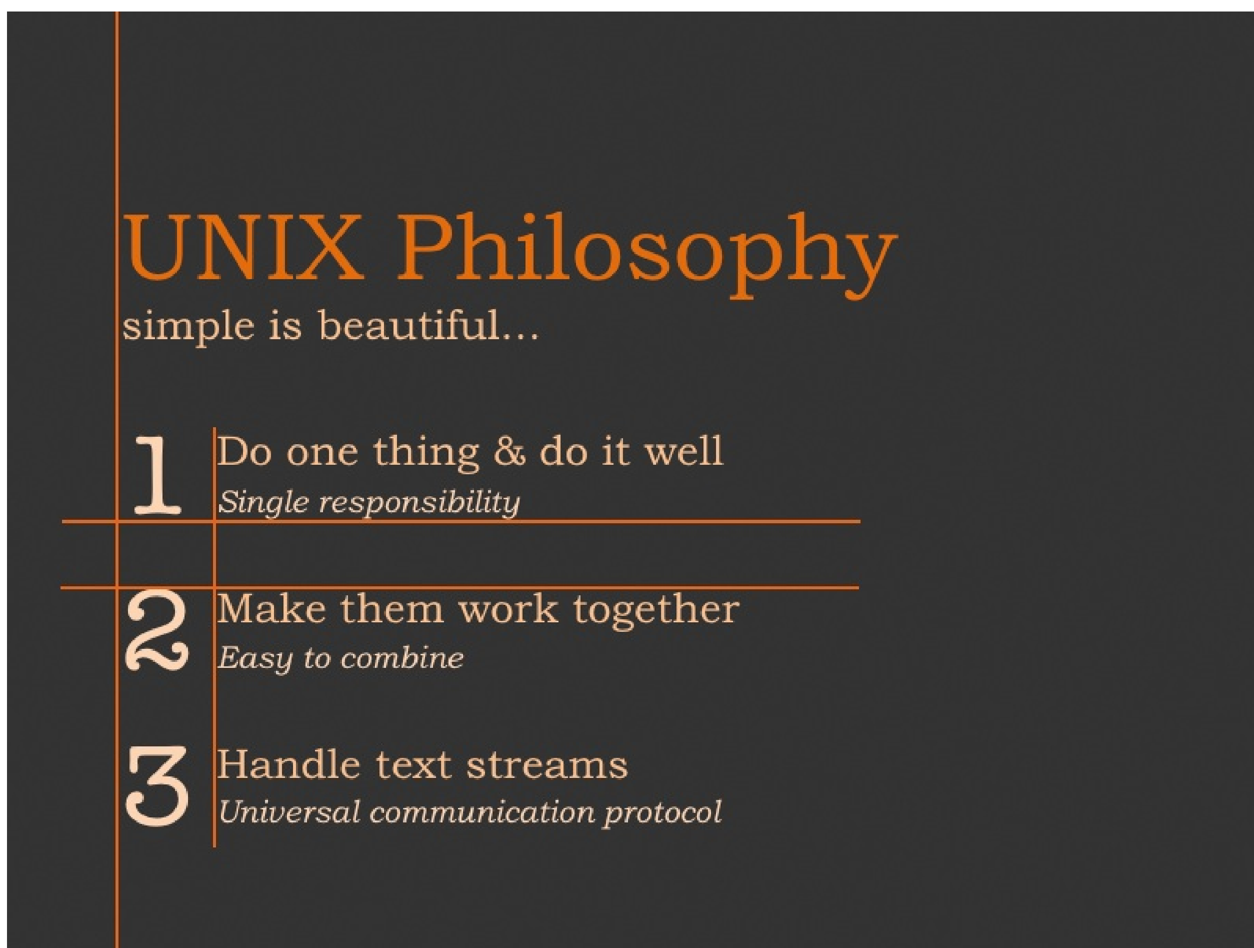
对齐

对齐是最容易做到，也最容易看到效果的一个原则。事实上，如果观察足够细致，你会发现这个原则在任何一个设计中都会存在。

我们来看一个设计，这是我做的一个关于`UNIX哲学`的演讲稿的设计中的一页。



这个设计中，所有的文本都遵循着一些看不见的线，并按照这些线来对齐：

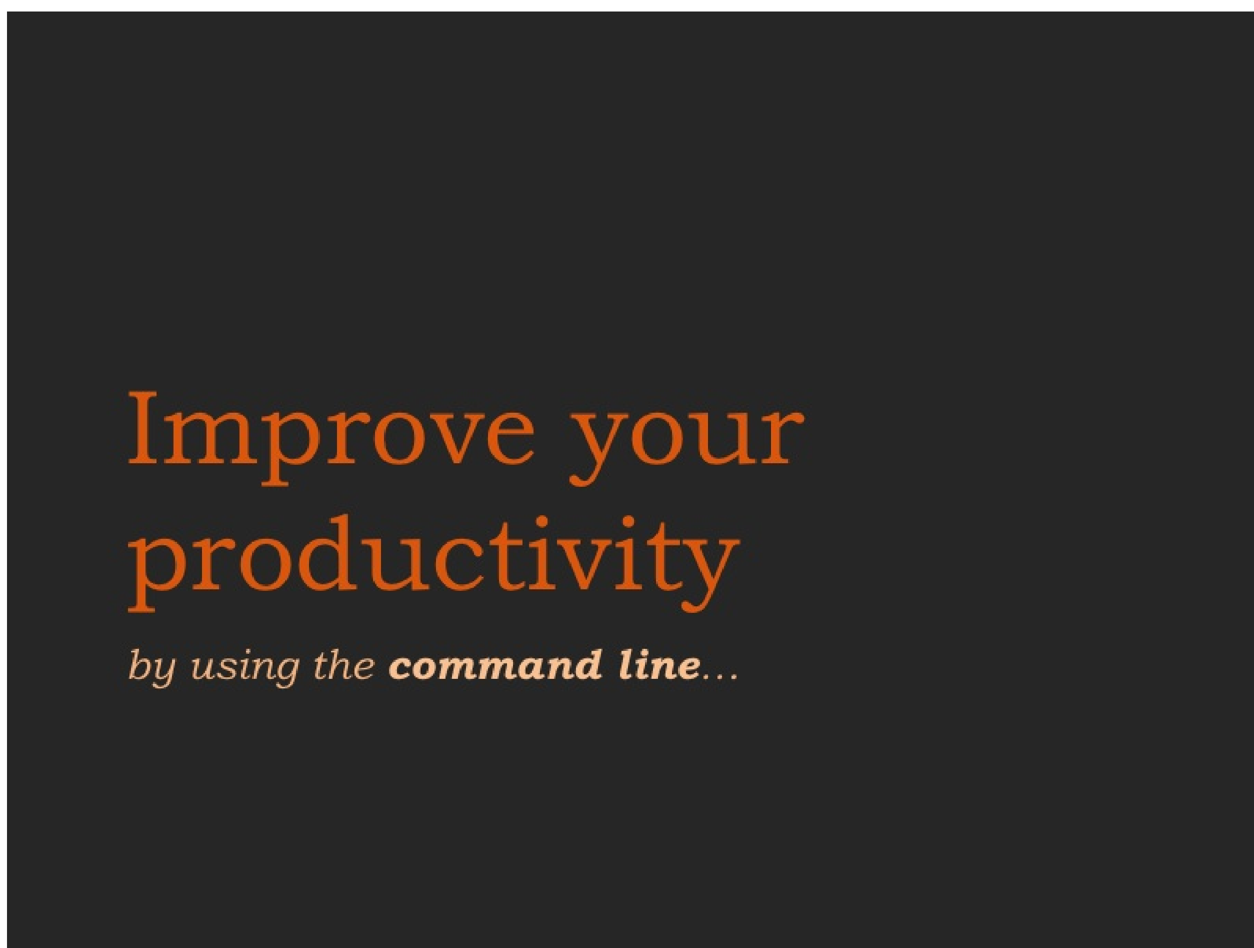


相比于凌乱，人们更倾向于阅读排列整齐的信息。如果图片，文本等的排列不整齐，用户需要花费一些精力来在大脑中对其进行处理，无形中就增大了使用的难度。当然，并非所有情况都需要严格的对齐，当你熟练掌握了设计的基本原则之后，就可以在适当的事后打破他们，比如下面这个设计就没有遵循对齐，但是却并没有影响到用户的体验：



对比

对比，即将不同的事物区分开来。比如前景色和背景色的区分，标题与正文的区分等等。在使用对比时，一定要足够强烈，比如使用互补色来进行对比，完全不同的两种字体来体现差异等。



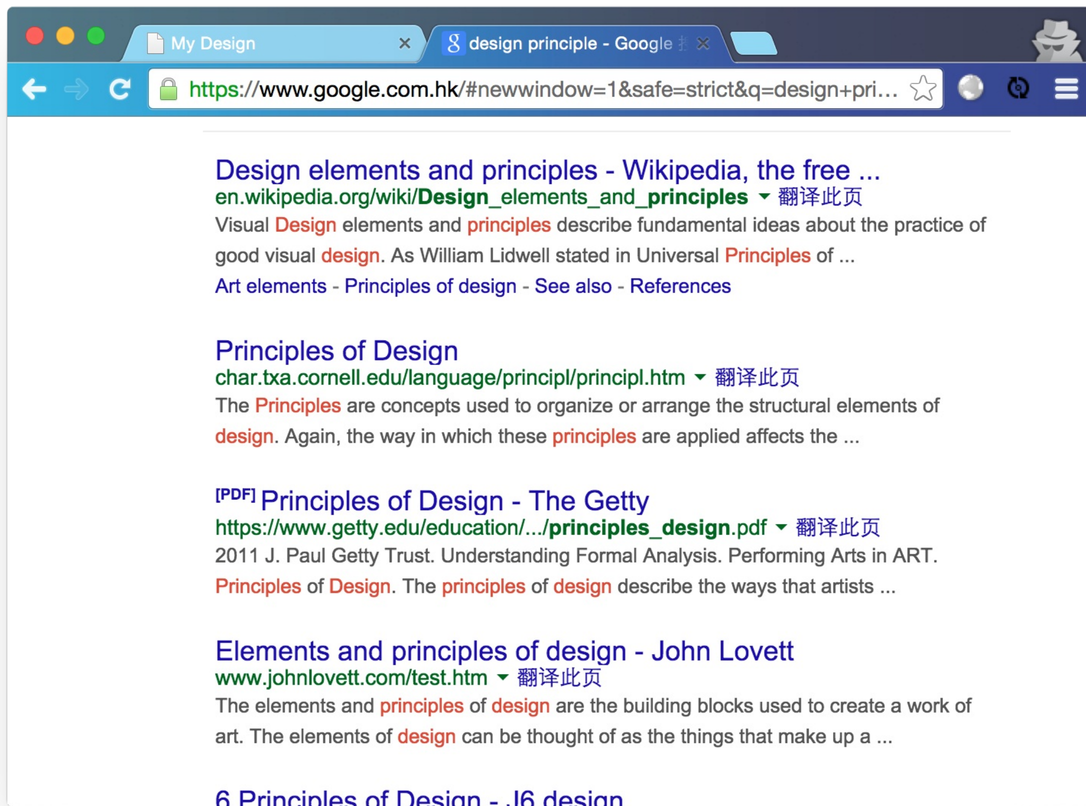
在上图中，我们可以看到文字的橘红色和背景的灰色形成了鲜明的对比。而主标题和副标题之间的文字大小，颜色的变化也可以让读者体会到两者的截然不同，最后，在副标题中，我们使用了斜体字来突出了 `command line` 这个关键字。

在对比中，事实上可以通过很多不同的方式来产生，常见的一些手法如下：

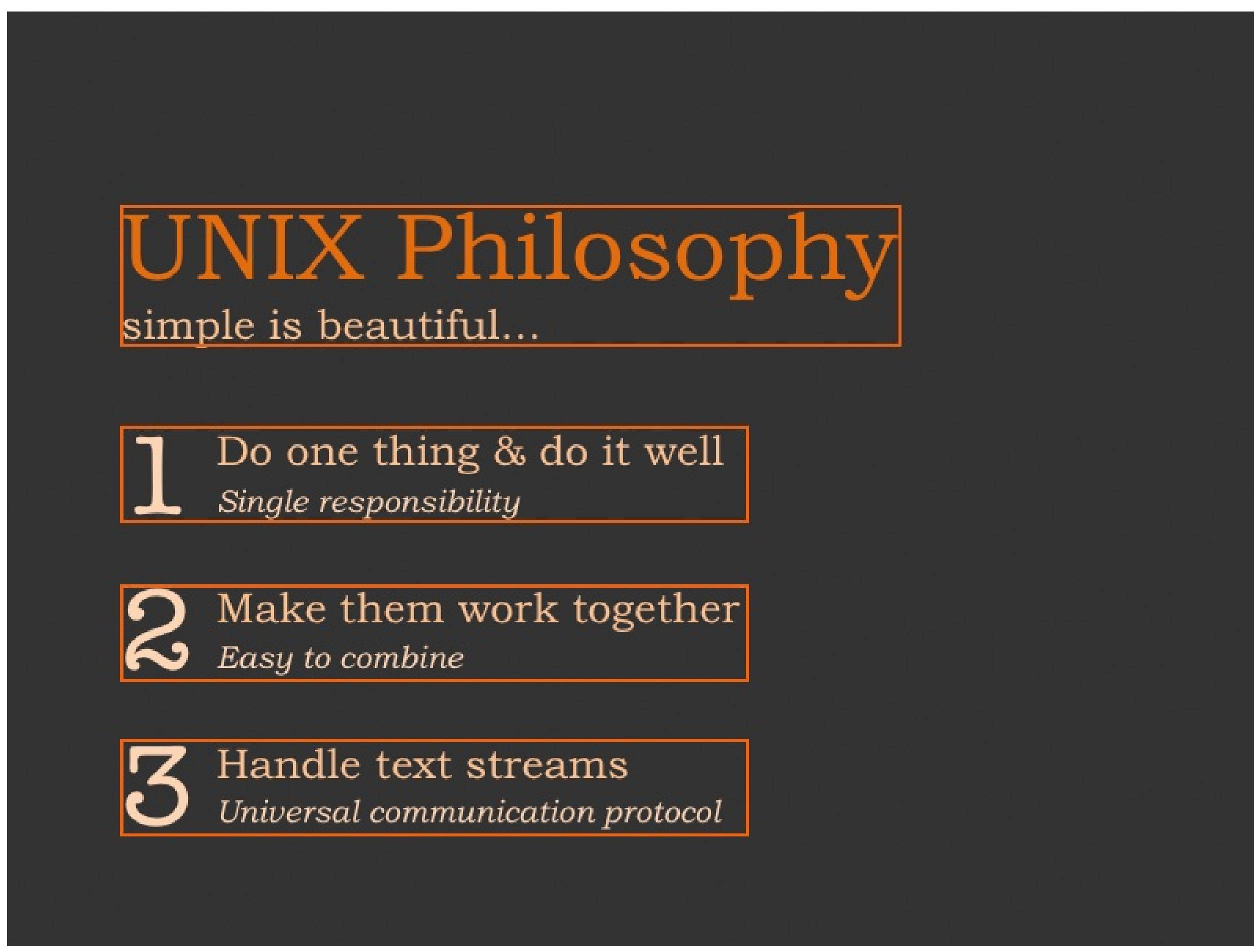
1. 颜色的不同
2. 不同类型的字体
3. 字体的大小
4. 字体的权重（粗体，常规，细体）

亲密性

人们习惯上会将距离较近的事物归为一组，或者至少认为它们 `具有相关性` 。比如我们在 `google` 中进行搜索，搜索结果界面设这样的：



我们事实上可以很容易的通过文字之间的空白大小来确定哪些文字属于第一个条目，哪些文字属于第二个条目。这就是亲密性原则的一个应用实例，如果我们再来分析一下`UNIX哲学`的那个例子，也可以看出亲密性原则的使用：

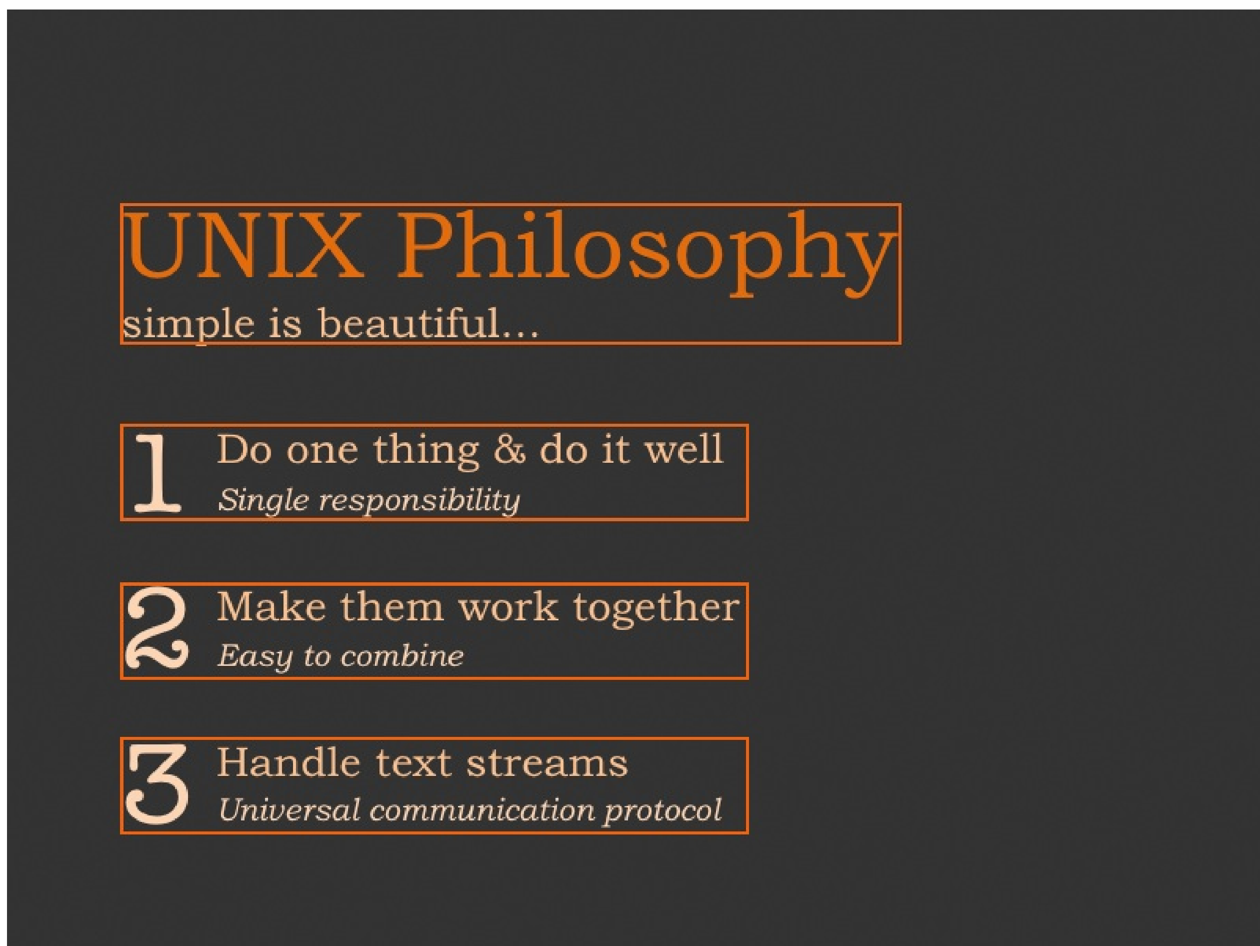


这个原则非常重要，我们将会在自己的设计中大量使用它。比如一大块信息和另一块之间留白的大小，标题和文本之间留白的大小，图片和描述文字之间的距离等等。使用亲密性原则，可以让节省读者很多的时间，从而让我们的设计更加有意义。想象这样一个场景：你正在通过手机浏览一个很长的列表，列表中的每一项都包含了一个图片和对应的描述文字，而你现在浏览到了第7个，但是由于所有的间距都一样，你就无法知道现在看到的描述文字是对上一张图片的描述，还是下一张。

重复

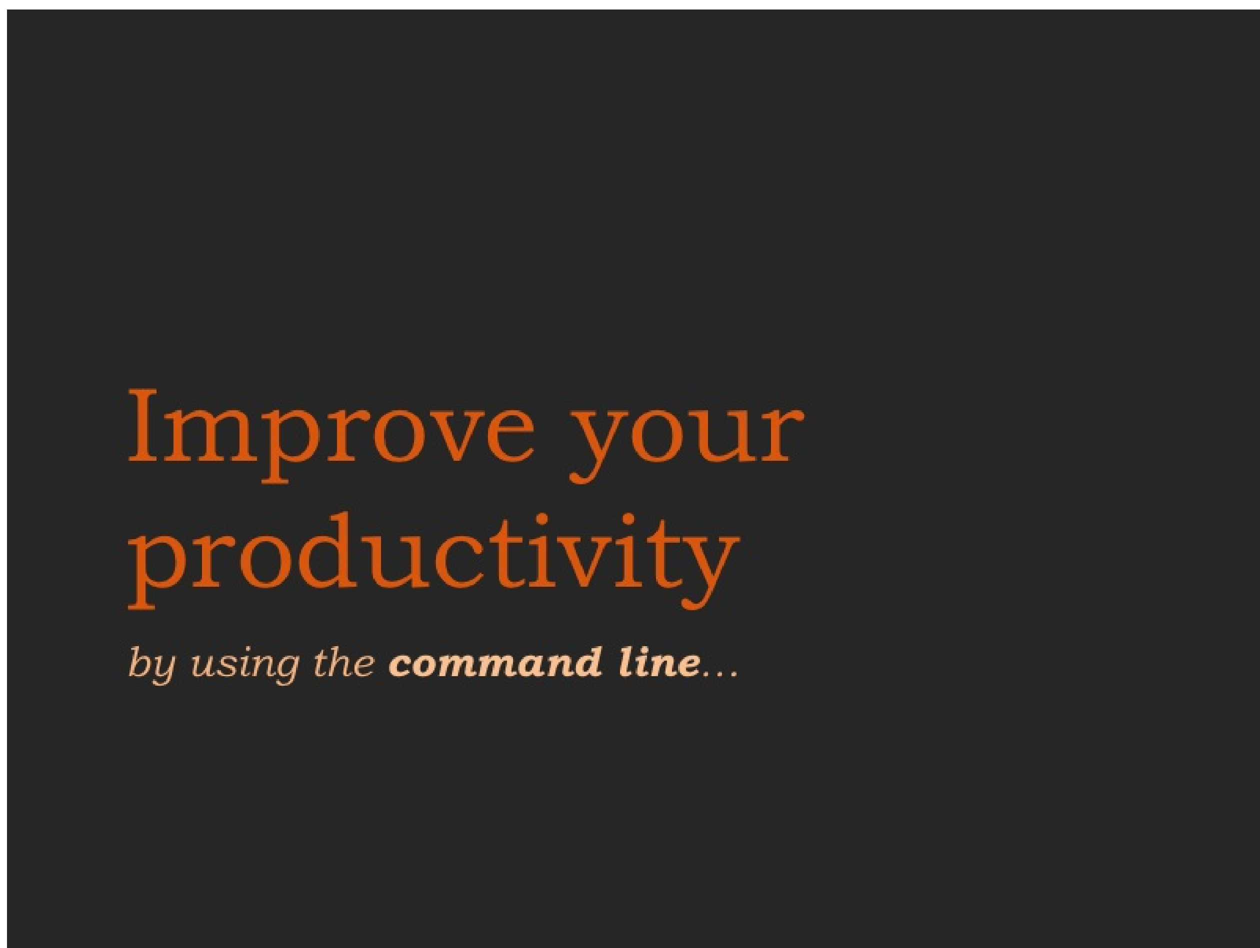
重复，即将一种模式不断使用。在设计中，有很多使用重复的地方，比如一个站点的高亮色，按钮/链接的样式。在同一个页面中有很多的重复，而在一个站点中，页面之间同样可以保持重复。

在亲密性原则中，我们看到的这张图



事实上也有包含了重复原则，比如分别列出的3个`UNIX哲学`，每个条目都包含了一个大的数字，然后一个主标题和一个副标题，而且主标题和副标题的字体还不一样。这种重复会形成一种节奏，当读者/用户使用这个设计时，在任何时候都不会觉得意外。即使这个设计是他第一次接触，当学会或者理解了第一个模式之后，后续出现的无非是第一个模式的重复。

比如这个设计中的另一个页面



你可以直接的感受到他们来自于同一个设计。现在我们来做一个小小的修改，将主标题和副标题的颜色做一下对调：

Improve your productivity

by using the ***command line***...

这就破坏了这种重复性，用户需要花费一点时间来适应这种调整，从而意识到我们的设计。注意，我们的目标是让用户感觉不到设计本身的存在。

如何组织内容

层次

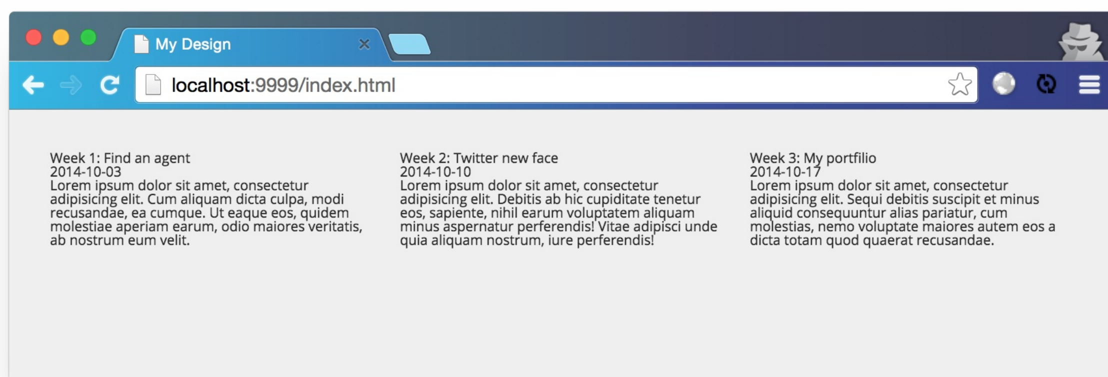
同样的内容，通过不同的方式传递出来的信息可以是`天差地别`的，我们这里有一个非常简单的例子：

假设我们为《3周3页面》做了一张课表，课表的内容如下：

```
<section class="schedule">
    <ul>
        <li>
            <div class="item">
                <h3>Week 1: Find an agent</h3>
                <time datatime="2014/10/03">2014-10-03</time>
                <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Cum aliquam
            </div>
        </li>
        <li>
            <div class="item">
                <h3>Week 2: Twitter new face</h3>
                <time>2014-10-10</time>
                <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Debitis ab h
```

```
                </div>
            </li>
            <li>
                <div class="item">
                    <h3>Week 3: My portfilio</h3>
                    <time>2014-10-17</time>
                    <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Sequi debiti
                </div>
            </li>
        </ul>
    </section>
```

如果使用浏览器默认的样式，上面的课表看起来是这样子的：



可以看到，课程名称，时间，课程描述的信息混合在了一起，读者需要花费一些精力（虽然可能仅需要数秒时间）才能找出自己关系的信息。

但是如果将不同权重的信息用不同的方式变现，则结果会大不相同:



显然修改之后的图显得更加易懂，读者不需要花费任何的思考就可以获得足够的信息。

颜色

颜色在很多时候都会影响使用者的情绪，比如红色代表热情（太阳，花朵等），蓝色表示安静平和（蓝天，大海的颜色）。因此在颜色的选取上也是需要仔细分析，仔细研究的。比如如果你为一家餐厅设计站点，那么红色可能是一个好的选择，但是如果你为一家航空公司或者安全相关的公司设计，则需要考虑蓝色或者绿色。

初学者的一个误区是使用多种颜色来组成自己的色彩方案。事实上，`Marketo`公司做过一次各个行业的公司在品牌颜色选择上的调查，结果非常有意思：有95%的公司仅仅使用了2种以内的颜色。而使用的最多的4中颜色依次为：蓝色，红色，黑色/灰色，黄色/金色。

在你自己进行设计时，如果不太确定到底要使用那种颜色的时候，不妨尝试蓝色。在选定主色调之后，就需要确定文字颜色和背景颜色了。事实上，文字颜色和背景颜色的选取和主色调可以完全无关。唯一的一条原则就是`保证易读性`。

比如：浅灰色的背景和深灰色的文字就是一个合理的搭配：



或者反过来的搭配也可以：



这两个例子中，文字和背景的对比都足够强烈，从而有较高的可读性。

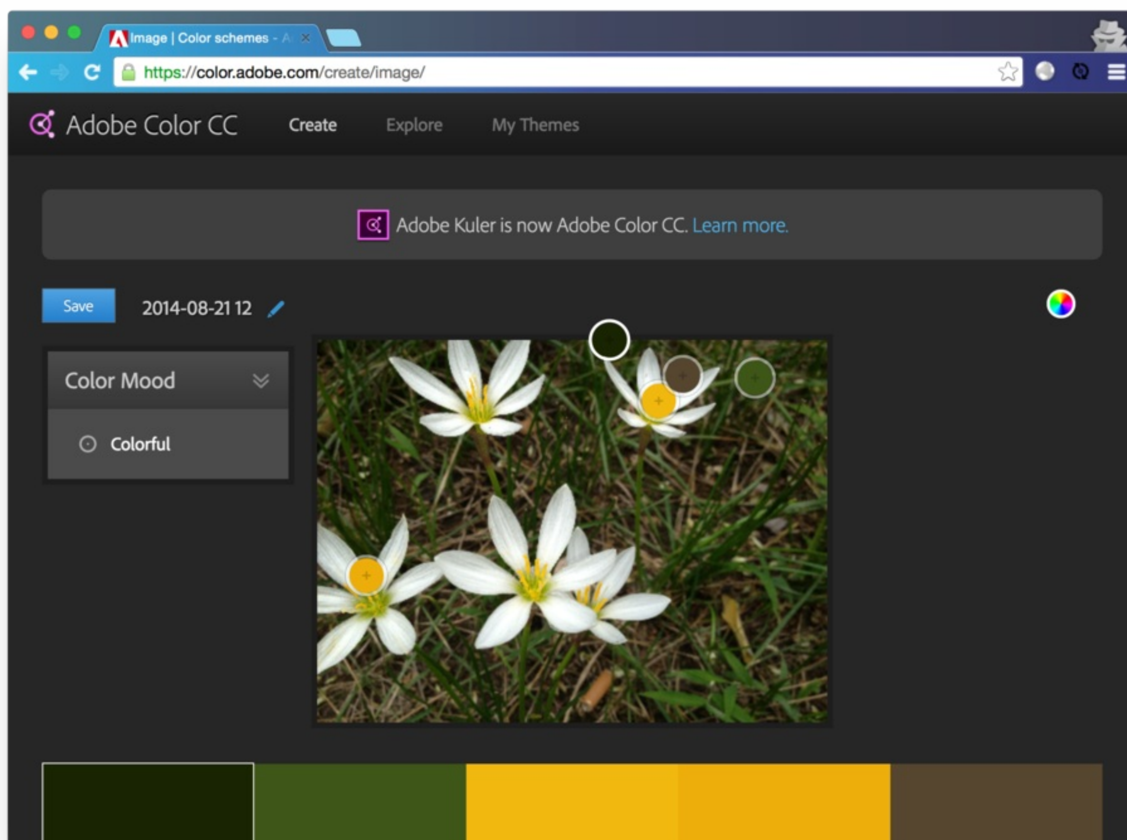
选取合理的色彩方案

要从众多的颜色中挑选出色彩方案从来都不是一件容易的事儿。但是有一些工具可以帮助我们自动完成挑选的动作，我们可以使用Adobe Kuler来根据照片选取颜色。

首先你需要找到一张与主题色彩相近的图片（不一定要跟主题内容相关，这里只关注颜色）：



然后将这样图片上传至Adobe Kuler即可：



你可以根据Kuler的一些选项来获得不同的颜色值，从而更加容易的找出一组合理的颜色方案。

图片

首先需要注意的是，图片本身就是内容的一部分。因此不要为了添加图片而使用图片，页面上的每张图片都需要有一个原因而存在。在页面中引入图片时，需要考虑这样一些因素：

1. 使用大而且清晰的图片
2. 使用与内容相关的图片
3. 图片的质量，尺寸，曝光等都非常重要
4. 考虑如何与文字混排

Hero Image

Hero Image是指一个站点上`黄金位置`的那张巨大的图片，这张图片通常都有很高的质量，而且和站点本身有很强的相关性。比如一些设计类的网站上，通常会有很整洁的桌子，上边整齐的摆放了一些工具。



一个开发工具网站，或者某个程序员的博客上，可以使用这样的Hero Image

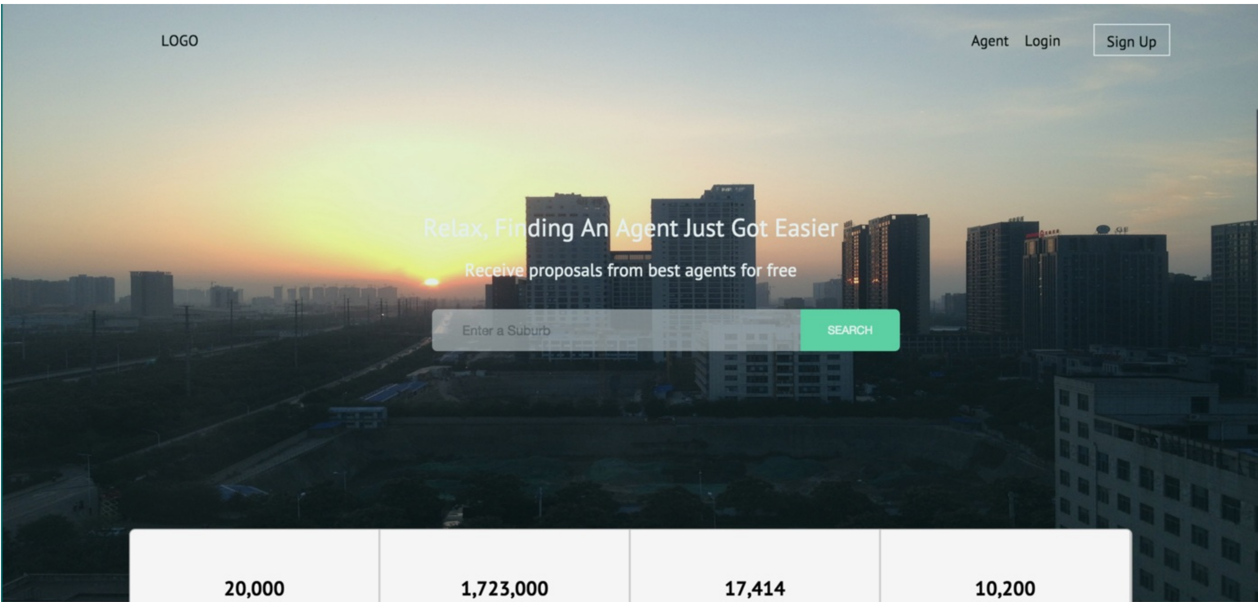


应该注意的是，Hero Image除了大而且清晰以外，需要保持与内容高度一致。一个精美但是与主题无关的

Hero Image反而会分散用户的注意力，产生负面的影响。

图文混排

通常我们会将图片和文字混合在一起，两者互为补充，相得益彰。在图文混排的时候，同样有一些原则需要注意，比如文字的颜色和图片本身的颜色过于接近，会造成阅读的困难等。



上图中的夕阳光线太强，而文本的颜色又是白色，两者混合之后就看不清了。

文字位置

文字的位置也非常重要，如果强行的将文字放置在图片中的主体之上，会产生一些负面的效果。我们来看一组对比：



上图中，文字覆盖在图片的主体（含羞草）之上，一方面会造成阅读的不便，另一方面又会阻止读者欣赏含羞草本身。如果将文字放置在`失焦`的位置，则两者都得到了足够的体现：



文字大小

在进化过程中，人类的眼睛对大的东西更加敏感（一头狗熊相对于一只田鼠，或者一只蜜蜂的威胁更大，因此也更加容易被眼睛关注到）。当映入眼帘的物体细小到一定程度时，它就很容易被过滤掉。

我们同样看两张用作对比的图片：



上图中，由于文字本身和背景图中的树叶大小相当，因此很容易被读者忽视。而如果将文字放到到一定尺寸，超过了背景图中的最大主体之后，它就会非常醒目：



而且文字和图片的对比还附有立体层次：小的文字离读者较远，而大的文字离读者较近。

图片亮度

通常情况下，人们拍摄的图片都是`曝光过度`，也就是说，白色太多。这在图文混排时，会给我们带来一定的麻烦。一个简单的处理方法是给图片蒙上一层灰色的层，然后在层之上编辑文字。



没有灰色图层之前，很难在图片上加入文字。



有了灰色图层之后，我们就可以加入亮色的文字了。

第二天

好了，我们已经学习了足够多的关于设计的理论了，让我们开始实际的开发吧。我们第三周的`作业`并没有任何的参考，需要我们自己从头开始设计，从主题的选择，Hero Image的使用，到图片的调整，颜色的搭配等，全部都由我们自己决定。

主题的选择

我在ThoughtWorks组织《3周3页面》这个Workshop的时候，参与者被要求设想一个自己最感兴趣，最愿意投入热情的主题，然后将这个主题转换为一个设计。如果实在想不出来，他们可以自己做一个漂亮的简历。

而我作为组织者，则希望做一个设计，将这个活动告诉给那些没有参加的其他同事。因此，我们这里的主题的选择比较容易，就叫《3周3页面》。有了主题之后，就可以考虑我们希望把什么样的信息传递给读者了。

内容

在这个设计中，我们需要传递给用户的是：

1. Workshop的主要内容
2. 提炼出这个Workshop一些独一无二的特点
3. 参与者如何学习
4. 一些活动的照片
5. 参与者的作品集
6. 必要的联系方式，如果读者对这个活动有兴趣，可以联系我们

应该注意的是，内容是一个站点与另外一个站点最`不同`的地方，设计本身是可以被复制的，但是内容不能（内容的复制就是抄袭，是最被人们所无齿，唾弃的行为之一）。

色彩选择

我喜欢简单的设计，因此主色调只需要一个。而根据主色调可以衍生出来一些同一个色系的好几个颜色。而文本内容和背景，则使用最经典的浅灰色背景，深灰色文字的搭配。

一个直观的用户界面指导是这样的：

Typography

H1 - Main title

H2 - Subtitle, sometimes in **bold**

H3 - Content headline

Body Text

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Buttons

Normal

Primary button

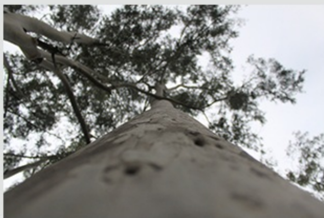
Hover

Primary button

Active

Primary button

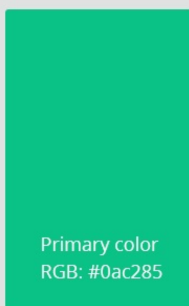
Images

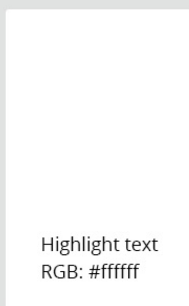


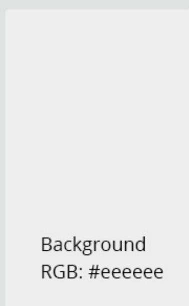
Avatars

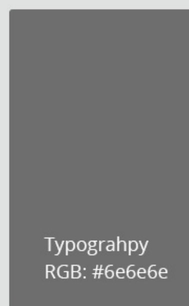


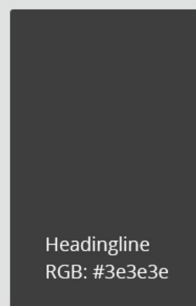
Colors











这样一个指导中，可以看出如何使用 `Heading`，按钮的颜色，鼠标移过按钮时候的颜色等。通常设计师应该提供这样一个 `UIKit` 供开发者参考（在我们的场景中，我们既是设计师，又是开发者）。

实现

选择Hero Image

Hero Image的选择上，我们希望找一张能体现团队合作，又能体现出活动相关内容（代码片段，编辑器等）。我在大家写代码的时候，拍摄了一些照片，其中这样我觉得和主题非常匹配：



导航栏

我们的设计是一个单页面设计，并不需要复杂的交互，在导航上，只需要让用户可以找到相关信息即可。我们这里仅仅需要两个链接：

```html
<header>
    <ul>
        <li><a href="https://github.com/abruzzi/3-pages-in-3-weeks">Code</a></li>
        <li><a href="http://icodeit.org/about-me/">About</a></li>
    </ul>
</header>
```

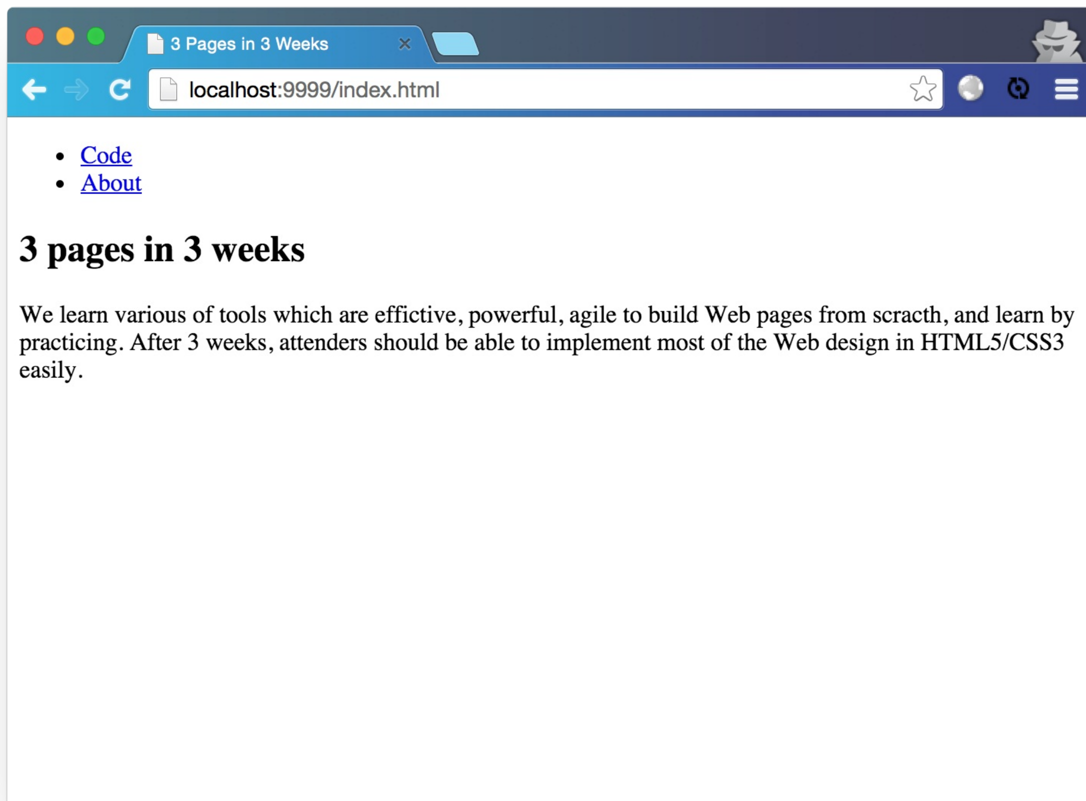
第一个可以导航到实际的代码（开发者可以参考我们的实现），另一个是关于作者的信息页面。

Headline

在Hero Image之上，我们需要一个 `Headline`，它需要足够简练。另外还需要对 `Headline` 的描述信息，

对应的HTML是这样的：

```
<section class="hero">
    <h2>3 pages in 3 weeks</h2>
    <p>We learn various of tools which are effictive, powerful, agile to build Web pages
</section>
```



接下来我们为这个区域加上样式，先来定义一些常量：

```
$background-color: #eeeeee;
$text-color: #3e3e3e;
$light-text-color: #eeeeee;
$heading-color: hsl(160, 90%, 40%);

$mask-color: #333333;
```

然后，在 `Body` 上定义一些通用的样式：

```
body {
    font-size: 62.5%;
    font-family: 'Open Sans', sans-serif;
    text-align: center;
```

```
    color: $text-color;
    background-color: $background-color;
    width: 100%;
}
```

紧接着是导航 `header` 和 `hero` 区域：

```
header {
    padding: 0 1em;
    background-color: $heading-color;
    li {
        float: left;
        a {
            font-size: 1.5em;
            text-decoration: none;
            color: white;
            display: inline-block;
            padding: 1em 1em;
        }
        &:hover {
            background-color: hsl(160, 90%, 35%);
        }
    }
    @include clearfix;
}

.hero {
    width: 100%;
    min-height: 50em;
    position: relative;
    background-color: $mask-color;
    z-index: 1;

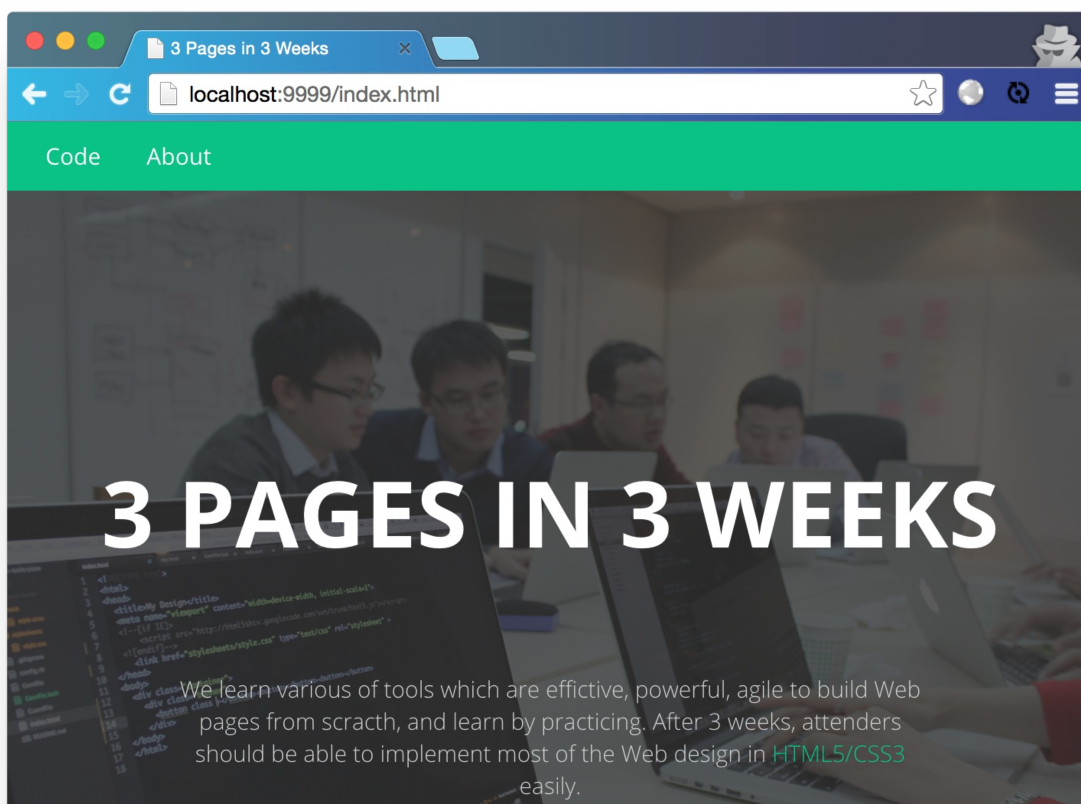
    &:after {
        background: url('../images/hero-1-resized.jpg');
        background-size: cover;
        position: absolute;
        content: "";
        z-index: -1;
        opacity: .2;
        width: 100%;
        height: 100%;
        top: 0;
        left: 0;
    }

    h2 {
        font-size: 6em;
        font-weight: bold;
        padding: 3em 0 1em 0;
        text-transform: uppercase;
        color: white;
    }

    p {
        max-width: 70%;
        font-size: 1.5em;
```

```
        font-weight: lighter;
        color: #cecece;
        line-height: 1.4;
        margin: 0 auto;
        padding: 1em 0 8em 0;
        span {
            color: $heading-color;
        }
    }
}
```

此处应该注意的是对于 `.hero` 这个section，我们将其背景设置为灰色，并将 `position` 设置为 `relative` 。然后为其添加了一个伪元素，这个伪元素上包含了实际的图片，并为图片设置了透明度 `.2` 。这样做的原因是如果直接设置 `.hero` 的透明度的话，其子元素 `h2` 和 `p` 都会被这个透明度影响。通过引入一个伪元素，我们就可以分别为文字和图片设置不同的透明度了。



特性区域

还记得在第一周的练习中， `Find an agent` 中的 `1,2,3` 3个步骤令人印象深刻，我们可以参考这个设计，将我们课程的亮点也用同样的形式展现出来：

```
<section class="topics container">
    <h3>What do we learn?</h3>
    <ul>
        <li>
```

```
            <div class="subject">
                <i class="icon-wrench"></i>
                <h4>Tools</h4>
                <p>We introduce tools like Sublime, Emmet, LiveReload, Guard to speed up
            </div>
        </li>
        <li>
            <div class="subject">
                <i class="icon-books"></i>
                <h4>Design theory</h4>
                <p>Some basic theory of design, color scheme, hierachy, imagery.</p>
            </div>
        </li>
        <li>
            <div class="subject">
                <i class="icon-circle-o-notch"></i>
                <h4>Workflow</h4>
                <p>A much more modern way of development, design, or both. We intend to m
            </div>
        </li>
    </ul>
</section>
```

由于我们之前已经实现过一次了，这样的样式我们可以很快写出来：

```
.topics {
    padding: 2em 0 4em 0;
    h3 {
        @include section-title;
    }

    li {
        float: left;
        width: 33.33%;

        i {
            font-size: 3em;
        }

        h4 {
            font-size: 1.5em;
            padding: 1em 0;
            font-weight: bold;
        }

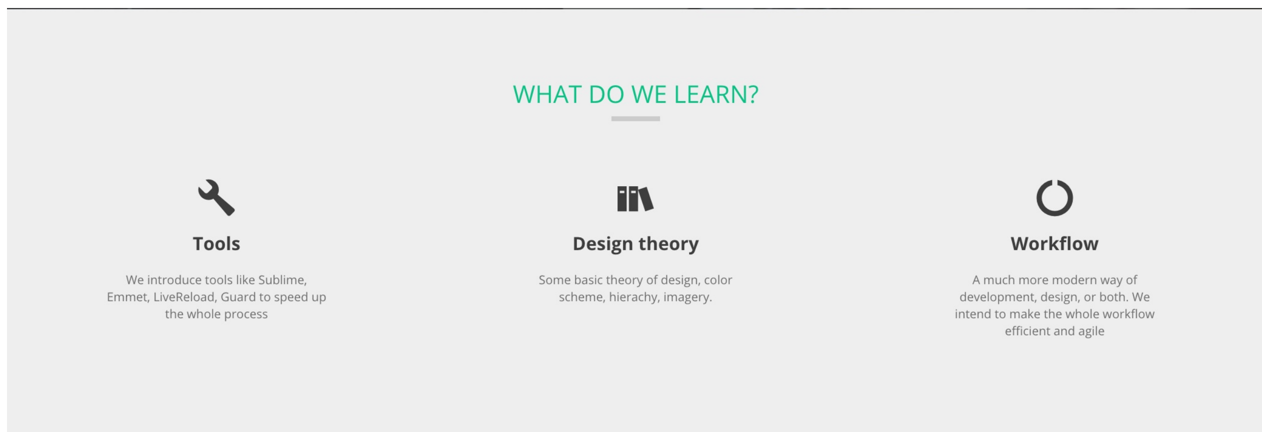
        p {
            line-height: 1.4;
            color: #6e6e6e;
            padding: 0 8em;
            margin-bottom: 4em;
        }
    }

    @include clearfix;
}
```

`section-title` 这个 `mixin` 的实现如下：

```
@mixin section-title() {
    font-size: 2em;
    text-transform: uppercase;
    color: $heading-color;
    padding: 2em 0;

    &:after {
        content: "";
        display: block;
        background-color: #cccccc;
        width: 2em;
        height: .2em;
        margin: .4em auto;
    }
}
```



样例区域

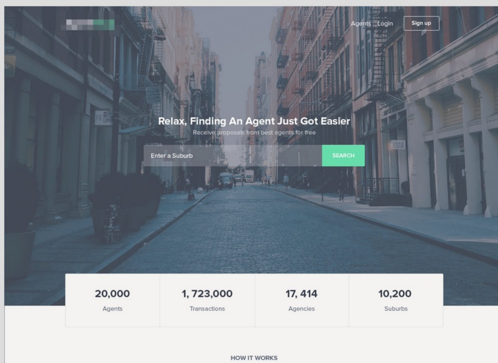
在计划内容的时候我们讨论过，要将参加者做的样例页面展现出来。样例包括一张截图，一个标题和一段描述文字，这样可以让读者直观的看到我们学习的样板是什么。

```
<section class="sample" id="week-1">
    <ul>
        <li>
            <div class="thumb">
                <img />
            </div>
        </li>
        <li>
            <div class="description">
                <h4>Week 1: Find an agent</h4>
                <p>In the very first week, we will learn how to setup the most modern and
                </p>
                <p>
                During the workshop, people learn how to implement the mockup from scrath
                <p>
                Also, we highly recommended people work in pairs, then they can learn fro
                </p>
            </div>
```

```
        </li>
    </ul>
</section>
```

对于每一个样例，HTML片段都是一样的，当前是第奇数个样例时，把图片在左边，描述信息在右边；当前样例是第偶数个时则相反。

SAMPLES TO BE IMPLEMENTED



Week 1: Find an agent

In the very first week, we will learn how to setup the most modern and effective envrionment first. We use Sublime + Emmet plugin for HTML/CSS code composing, and Compass + SCSS for css, then Guard + LiveReload for quick feedback and enhancement.

During the workshop, people learn how to implement the mockup from scrath, how to structure the HTML document, how to float element, how to use pesudo elements, etc.

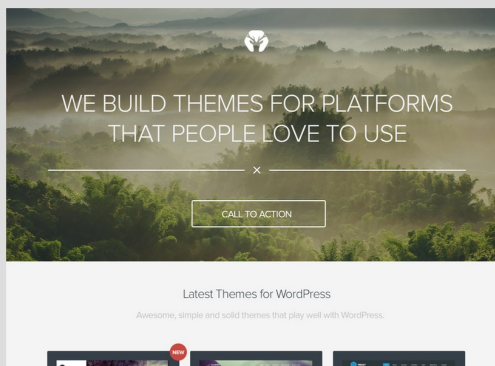
Also, we highly recommended people work in pairs, then they can learn from each other, and get comfortable when blocked by some details.

Week 2: Themes for wordpress

In this week, we are looking into HTML5 elements, and talked about semantics tags. Later on, we leant new features in CSS3, like transition, gradient and transform.

After that, we discussed some color related topic like RGBa and HSL/HSLa. Also leant how to choose your own color by just think in the HSL way.

People from different background are keeping pair and learn stuff from each other, more people are joining, we are running out of talbe.



这部分非常简单，样式如下：

```
li {
    float: left;
    width: 50%;

    .thumb {
        width: 100%;
        height: 30em;
        overflow: hidden;
        img {
            width: 90%;
            margin: 0 auto;
        }
    }

    .description {
        h4{
```

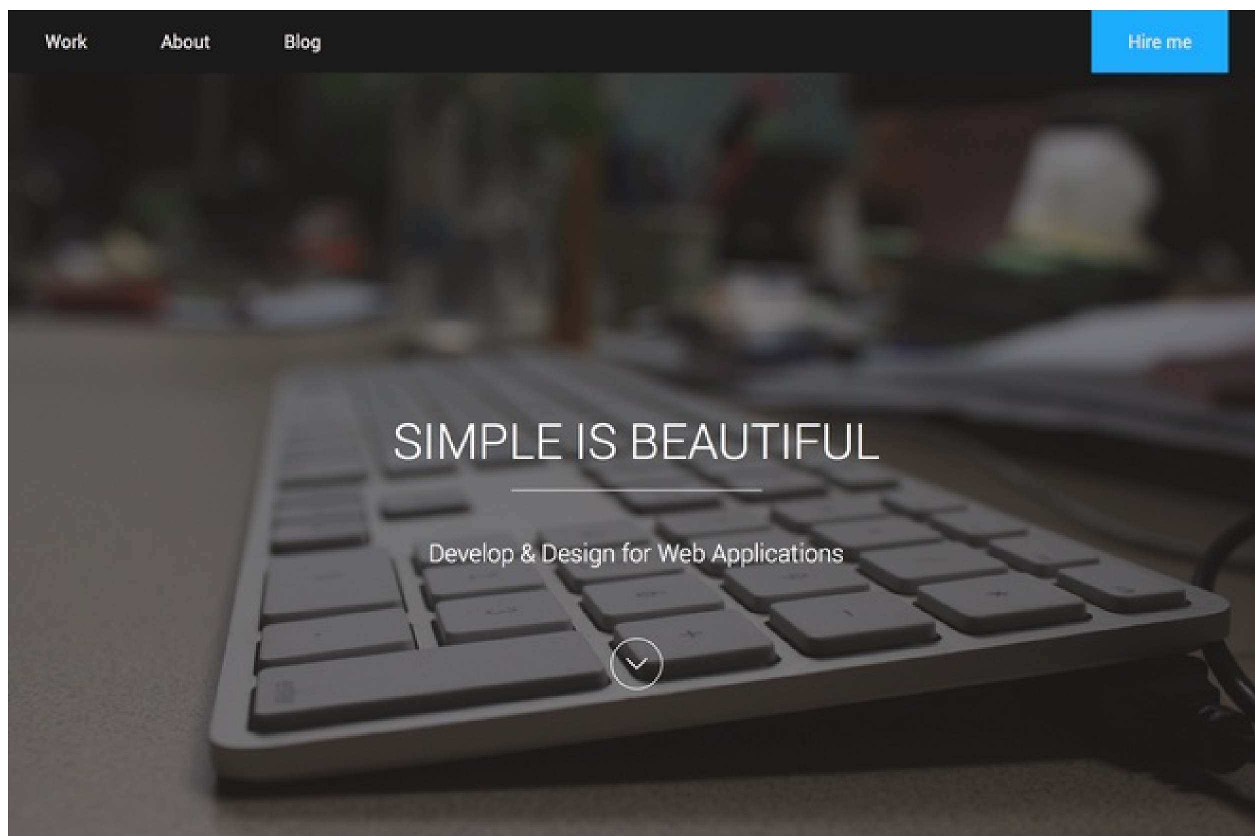
```
            text-align: left;
            padding-left: 1.5em;
            font-size: 2em;
            font-weight: bold;
        }

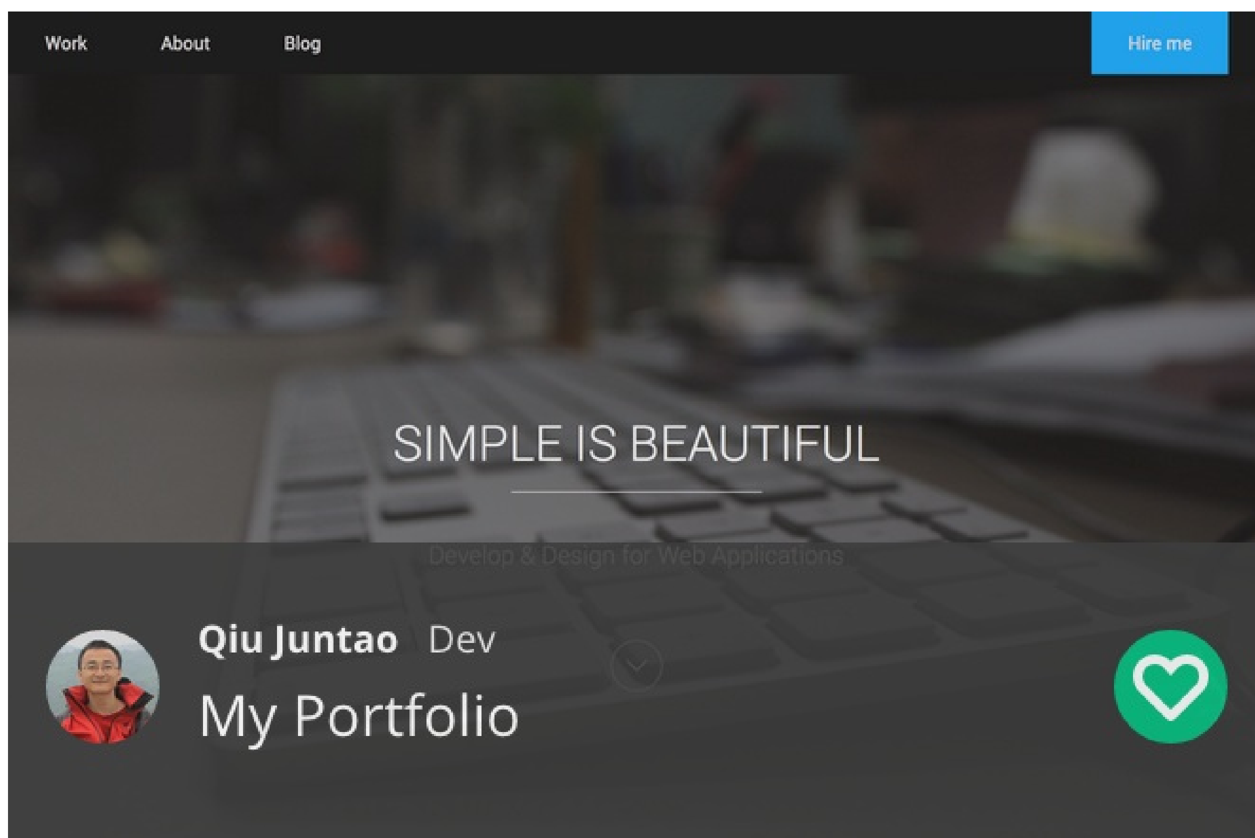
        p {
            font-size: 1.5em;
            font-weight: lighter;
            line-height: 1.4;
            color: #6e6e6e;
            text-align: left;
            padding: .5em 2em;
        }
    }

    @include clearfix;
}
```

展板

展板部分会展示所有参与者提交的作品，每个作品都包含一个截图，同时会有一些说明信息，比如作者的名称，角色，头像等。要组织并展示这些信息殊非易事，我们可以充分利用其它资源。比如我在浏览别的设计师的作品时发现，有一种做法是展板上仅仅展示静态的图片，而当鼠标移动到相关的设计上时，会滑出一个小的面板，上面包括了图片的说明，当鼠标挪开之后，这个面板又会消失。





要实现这种效果，我们先完成HTML的代码：

```
<ul>
    <li>
        <a href="http://icodeit.org/portfolio-timeline/" target="_blank" class="mask"></a
        <img src="images/qiujuntao-showcase.png" alt="" class="case">
        <article class="information">
            <section class="detail">
                <img src="images/qiujuntao-avator.jpg" class="avator">
                <div class="contact">
                    <span class="name">Qiu Juntao</span>
                    <span class="role">Dev</span>
                    <time class="date">2014-11-28</time>
                    <p>My Portfolio</p>
                </div>
                <a href="#" class="like">
                    <div class="icon-heart"></div>
                </a>
            </section>
        </article>
    </li>
    ...
</ul>
```

列表中的每一项都含有一个链接和一张图片，以及一个隐藏的`article`，这个隐藏的`article`会在鼠标挪到图片上时滑出来，并在鼠标挪走的时候又消失。

```
li {
```

```
    float: left;
    width: 33.33%;
    box-sizing: border-box;
    position: relative;

    height: 22em;
    overflow: hidden;

    .case {
        width: 100%;
    }

    &:hover {
        .mask {
            top:0;
        }
    }

    .mask {
        display: block;
        position: absolute;
        top:-22em;
        left:0;
        height: 100%;
        width: 100%;
        background-color: $text-color;
        opacity: .1;
    }
}
```

我们设置每个条目占用总宽度的33.33%，即每行可以放置3个作品。然后定义了 `mask` 的样式，这个 `mask` 就是条目上的a链接，为了扩大可点击区域，我们将整个图片都变成了可点击区域。这个 `mask` 的top开始设置为 `-22em` ，也就是隐藏起来，然后在 `hover` 时将这个值设置为 `0` 即可。

而对于 `information` 这个隐藏的块，我们定义了以下样式：

```
.information {
    position: absolute;
    background-color: $text-color;
    top:22em;
    left:0;
    width: 100%;
    height: 100%;
    opacity: .9;
    transition: all .4s ease-in-out;
    text-align: left;
    padding: 0 1em;
    box-sizing: border-box;
    color: white;

    h3 {
        font-size: 2em;
        padding: 1em 0;
    }

    .detail {
```

```
            position: relative;
            padding-top: 2em;
            .avator {
                position: absolute;
                width: 3em;
                height: 3em;
                bottom: 0;
                @include border-radius(100%);
            }

            p {
                padding: .5em 0 0 0;
                font-size: 1.5em;
            }
            .contact {
                margin-left: 4em;
                .name {
                    font-weight: bold;
                    margin-right: .5em;
                }

                .role {
                    color: $light-text-color;
                }
            }

            .like {
                position: absolute;
                font-size:2em;
                bottom: 0;
                right: 0;

                color: white;
                background-color: $heading-color;
                border: 1px solid $heading-color;

                padding: .2em;
                @include border-radius(100%);
                transition: all .3s ease-in-out;

                &:hover {
                    color: $heading-color;
                    background-color: white;
                    border: 1px solid transparent;
                }
            }
        }
    }
```

最终结果

Code About

3 PAGES IN 3 WEEKS

We learn various of tools which are effictive, powerful, agile to build Web pages from scracth, and learn by practicing. After 3 weeks, attenders should be able to implement most of the Web design in HTML5/CSS3 easily.

WHAT DO WE LEARN?

Tools

We introduce tools like Sublime, Emmet, LiveReload, Guard to speed up the whole process

Design theory

Some basic theory of design, color scheme, hierachy, imagery.

Workflow

A much more modern way of development, design, or both. We intend to make the whole workflow efficient and agile

SAMPLES TO BE IMPLEMENTED

Relax, Finding An Agent Just Got Easier

20,000 | 1, 723,000 | 17, 414 | 10,200

Week 1: Find an agent

In the very first week, we will learn how to setup the most modern and effective envrionment first. We use Sublime + Emmet plugin for HTML/CSS code composing, and Compass + SCSS for css, then Guard + LiveReload for quick feedback and enhancement.

During the workshop, people learn how to implement the mockup from scrath, how to structure the HTML document, how to float element, how to use pesudo elements, etc.

Also, we highly recommended people work in pairs, then they can learn from each other, and get comfortable when blocked by some details.

Week 2: Themes for wordpress

In this week, we are looking into HTML5 elements, and talked about semantics tags. Later on, we leant new features in CSS3, like transition, gradient and transform.

After that, we discussed some color related topic like RGBa and HSL/HSLa. Also leant how to choose your own color by just think in the HSL way.

People from different background are keeping pair and learn stuff from each other, more people are joining, we are running out of talbe.

WE BUILD THEMES FOR PLATFORMS THAT PEOPLE LOVE TO USE

CALL TO ACTION

Latest Themes for WordPress

"ARTISTS WHO JUST HAPPEN TO USE VIDEO EQUIPMENT"

CRAFT CREATIVITY CHARACTER

Week 3: Artists

In week 3, we changed the plan, and get started to do some design work. Attenders are required to design their own Web page, it could be any topic basically.

And people were showing creative and productive, they worked very hard and effecitvely

During the workshop, we also learn some basic design princples. We talked about how to use image, how to do basic typograghy, and hierachy.

All the design are all from internet, like dribble.com, or just found in pinterest.com. We don't actually have the copy right, we're using it just for learn how to implement a beautiful Web page. If this bother you, please contact me, and I'll remove it, thanks.

HOW DO WE LEARN

PAGE GALLERY

Resume

F.C. Barcelona

How to make Christmas biscuit?

SIMPLE IS BEAUTIFUL

PENGUIN'S LIFE

Relax, Finding An Agent Just Got Easier

Created by Juntao Qiu

动画

相较于静态内容，动态的内容对人眼更有吸引力。这可能来源于人类在漫长的进化中获得的能力，动态比静态更加危险！事实上，现实世界中我们已经不自觉的在使用这个法则，状态栏闪烁的QQ头像，界面上的弹出窗口，到处飘动的小广告，GIF动图，Flash广告等等。



页面上的动态元素会非常吸引读者的注意力，会使得页面更加生动，更清晰传神的表达出内容。我们常说，一图胜千言，而一个动画又可以胜过了多张图片（当然，我们需要把握好度，不能让动画干扰用户对内容的消费）。

页面大小

按照传统的方式，在页面上实现动画的方式有多种：

1. 使用GIF图片
2. 使用JavaScript来动态修改DOM的位置，加上一个定时器
3. 使用Flash

而这些方法都或多或少有些问题，比如引入额外的文件会导致页面尺寸变大，这样会导致页面加载变慢，从而影响用户体验。而CSS3的出现大大简化了动画的实现方式。通过浏览器对标准的支持，我们仅仅听过CSS就可以实现动画，而不依赖于大尺寸的外部文件。

基本上，CSS3通过过渡（transition），变形（transform）动画（animation）等方式来支持动画。我们可以分别来看。

过渡和变形

过渡

过渡是CSS3标准的一部分，用来控制CSS属性变换的速率。比如最常见的情况，我们通过 `:hover` 伪选择器来定义鼠标移到元素上的状态，我们为 `:hover` 状态定义了一些不同的属性值（不同的背景颜色，不同的字体大小等等），当鼠标移动到元素上之后，我们会看到属性值会被迅速的修改。

```
<div class="circle"></div>
```

比如我们为 `.circle` 定义了这样的样式，将它绘制成一个圆形：

```
.circle {
    width: 20em;
    height: 20em;
    border: 1px solid #c0c0c0;
    box-shadow: 0 0 5px #c0c0c0;
    border-radius: 50%;
    background-color: yellowgreen;
}
```

然后当 `:hover` 发生时，将该圆的背景改成橙色：

```
.circle:hover {
    background-color: orange;
}
```

过渡就是将这个过程延长，我们只需要定义其实状态和结束状态的属性值，然后浏览器会根据预设的时间间隔来自动计算过程中的属性值。



要定义一个过渡，我们可以指定这样一些参数：

1. 开始时间（何时开始动画）
2. 持续时间（动画持续时长）
3. 属性值根据时间的变化函数（线性匀速，非线性）

其语法为：

```
#element {
    transition: <property> <duration> <timing-function> <delay>;
}
```

比如我们的上例中，可以定义这样的过渡：

```
.circle {
    width: 20em;
    height: 20em;
    border: 1px solid #c0c0c0;
    box-shadow: 0 0 5px #c0c0c0;
    border-radius: 50%;
    background-color: yellowgreen;
    transition: background-color 1s ease-in-out 0s;
```

```
}
```

这行过渡定义表示，我们要过渡 `background-color` 属性，持续时间为1秒，时间函数为淡入淡出，开始时间为 `立即` 。

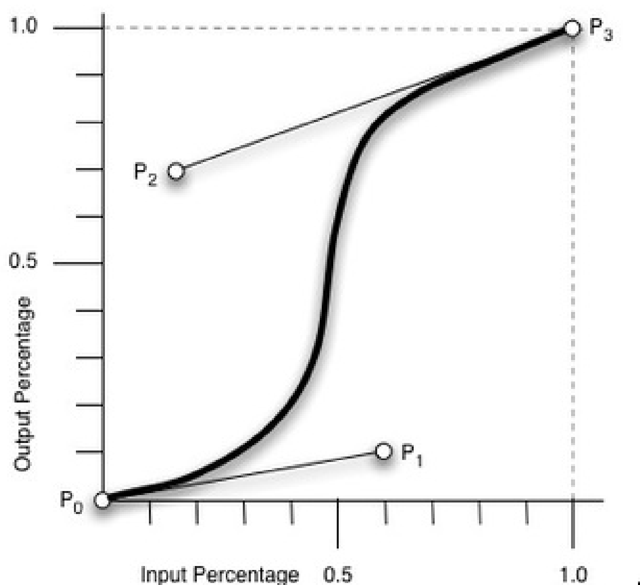
如果我们这里有多个属性要写，可以简写为 `all`：

```
transition: all 1s ease-in-out 0s;
```

如果无需延时执行，可以将此处的 `0s` 忽略。

时间函数是一个通过4个参数来定义的贝塞尔函数，简而言之，它定义了属性的值如何根据时间的变化而变化。比如匀速变化，先快后慢的减速变化，先慢后快的加速变化等等，CSS3已经预定义了一些命名的函数：

1. linear 表示匀速
2. ease 逐渐变慢
3. ease-in 加速运动
4. ease-out 减速运动
5. ease-in-out 先加速后减速



如果你精通数学的话，可以通过参数来自动以一个贝塞尔函数：

```
cubic-bezier(x1, y1, x2, y2)
```

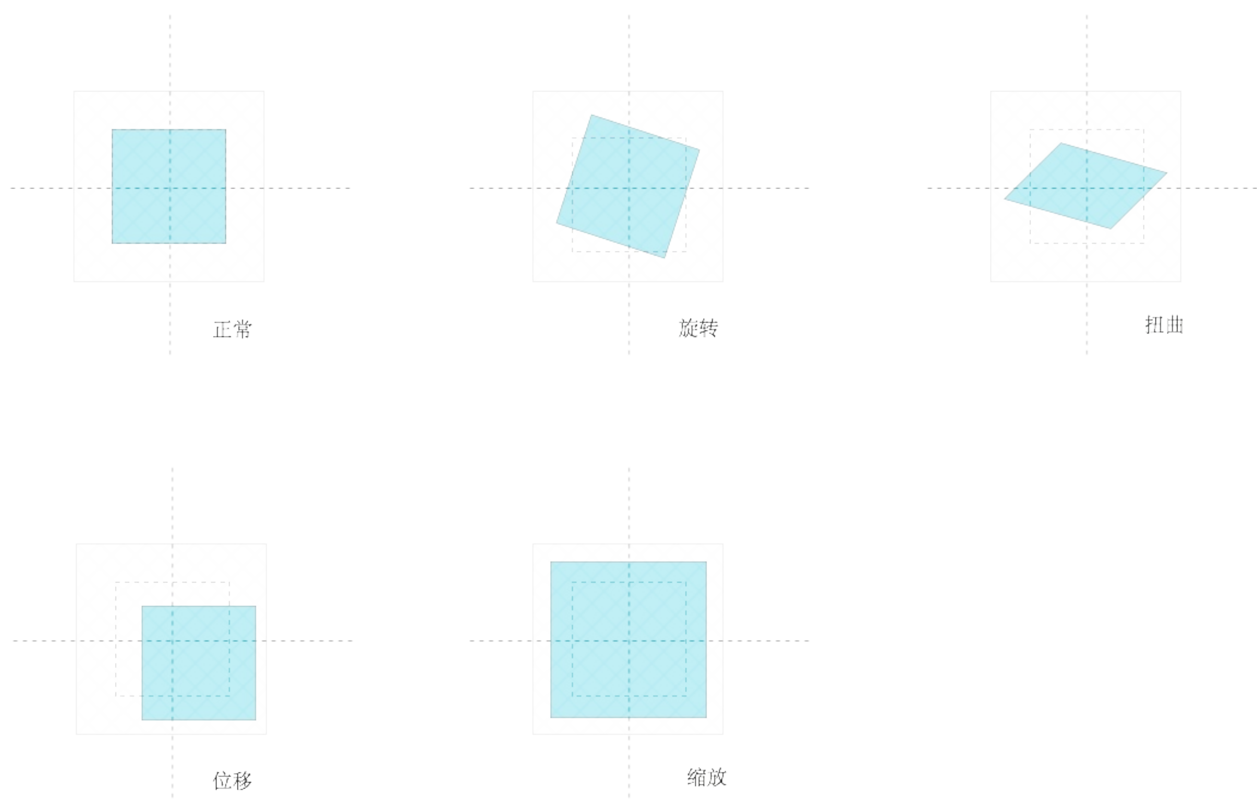
你还可以通过这个地址Cubic Bezier进行预览，然后将参数拷贝到自己的CSS代码中。



这样，当用户再将鼠标移动到圆上的时候，背景色会由`黄绿`慢慢的过渡到`橙色`。

变形

变形是另外一个非常重要的CSS3特性。变形是指将元素的尺寸，位置，形状通过函数来进行变化。目前包括旋转（rotate），扭曲（skew），缩放（scale），移动（translate）以及矩阵变形（matrix）。



旋转的语法为：

```
transform: rotate(<angle>);
```

`rotate(10deg)` 表示将该元素顺时针旋转10度，如果度数为负数，则表示逆时针旋转。比如：

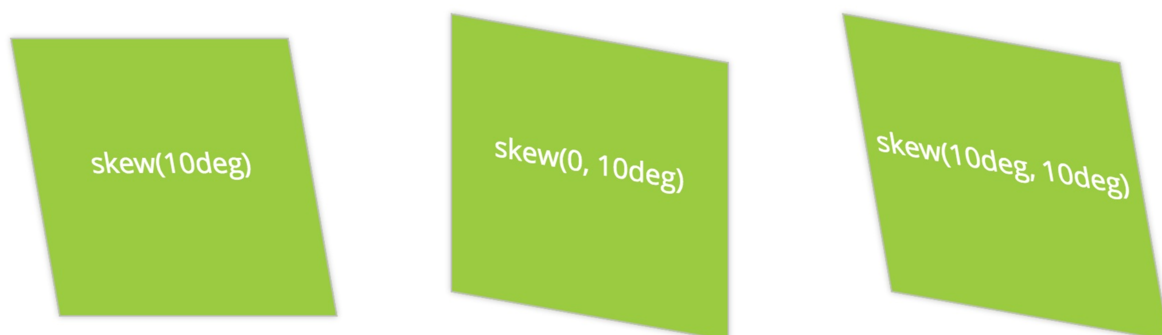
```
.square {
    margin: 0 5em;
    width: 20em;
    height: 20em;
    border: 1px solid #c0c0c0;
    box-shadow: 0 0 5px #c0c0c0;
    background-color: yellowgreen;
    transform: rotate(10deg);
}
```



扭曲会将元素按照x，y方向以一定的角度进行变形。这样可以将矩形变为平行四边形，菱形等形状。扭曲的语法为：

```
transform: skew(<x-angle>, <y-angle>);
```

```
.square {
    transform: skew(10deg);
}
```



缩放将元素变大，涉及到缩放的时候，有一个缩放比率的问题，即元素在x，y方向的缩放比例通常需要一致。当然有些情况下，我们需要元素某个维度保持不变，而仅仅修改另一个维度。缩放的语法为：

```
transform: scale(<number>[, <number>]);
```

这个数字是缩放的倍数，大于1表示放大，小于1表示缩小。第二个参数可以忽略，当忽略时，x和y都按照同一数字进行缩放。

位移可以将元素按照x，y方向移动。当然还可以两个参数同时指定。位移的语法为：

```
transform: translate(<x-value>[, <y-value>]);
```

比如：

```
transform: translate(10px, 10px);
```

将元素向x，y方向都移动10个像素。而且此数值可能为负，负数表示反方向移动。

`matrix`比较复杂，这里就不做深入讨论了。有兴趣的可以移步此处深入学习。

变形本身无法完成动画，但是当它和过渡结合起来之后，就具备了实现动画的一切条件。简而言之：变形定义了如何改变元素的形体，而过渡定义了时间关系。组合起来，我们就可以让一个元素，在某一段时间内，由一个形体慢慢的变成另外一个形体，这就是动画。

动画

其实有了上边的过渡和变形，我们就可以完成很多动画的动作部分。比如，当鼠标`:hover`到一个元素的时候，我们将这个元素水平移动300px，但是这个过程需要持续1秒：

```
.circle {
    transition: all 1s ease 0s;
}

.circle:hover {
    background-color: orange;
```

```
    transform: translate(300px);
}
```

这时候当移动鼠标到 `.circle` 上时，就会看到这个元素会缓慢的移动到右边。

我们还可以给 `:active` 状态加一个不同的样式：

```
.circle:active {
    background-color: orangered;
    transform: scale(1.2);
}
```

当用户按下鼠标（点击button的动作），会看到圆形会变大一些，而且颜色会变成 `橘红色`，当然由于设置了过渡，这个过程式渐变式的。

但是这个动画的也有限制：

1. 无法控制动画播放的时机
2. 无法让动画循环播放

这就需要我们引入更高级一些的动画（animation），首先我们来看看 `关键帧` 的概念

关键帧

关键帧在很多动画系统都有定义。关键帧中包含了一些自定义的状态，每个状态都会定义一些不同的属性值，然后在这些状态之间，浏览器会自动插入一些值（通过过渡的方式）。

比如，我们可以定义这样的一个关键帧：

```
@keyframes breath {
    0% {
        opacity: 1;
    }

    50% {
        opacity: .3;
    }

    100% {
        opacity: 1;
    }
}
```

这个关键帧的名字为 `breath`，状态 `0%` 的时候，不透明度为1，`50%` 的时候为0.3，`100%` 的时候又恢复到1。定义好关键帧之后，我们需要通过 `animation` 属性来使用它：

```
.circle {
    animation: breath 2s ease-in-out infinite;
}
```

我们指定`animation`的一些参数：使用那些关键帧，动画持续多长时间，以何种时间函数，动画重播次数等。这些参数可以分别指定：

1. animation-delay 从加载到执行动画间的延迟，默认无延迟
2. animation-direction 动画播放完成之后，重播的起始方向
3. animation-duration 动画播放周期时长
4. animation-iteration-count 重复次数
5. animation-name 关键帧名称

比如下面这个属性指定：使用`breath`关键帧，以2s为周期，时间函数为`ease-in-out`，然后无限循环播放。

```
animation: breath 2s ease-in-out infinite;
```

我们再来定义一个跳动的效果：

```
@keyframes beat {
    0% {
        transform: scale(1);
    }

    50% {
        transform: scale(1.2);
    }

    100% {
        transform: scale(1);
    }
}
```

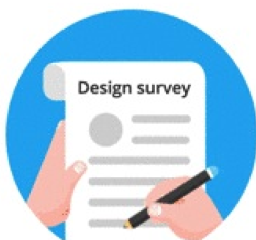
然后在某个元素中使用这个关键帧：

```
.circle {
    animation: beat 2s ease-in-out infinite;
}
```

实例

有了这些基础知识，我们就可以来进行一些实例的开发了。这里有一组非常有趣的动画，设计师提供的是GIF格式的图片。为了完成动画，我们首先需要设计师提供的素材集：所有动画元素的原图。

Design Survey



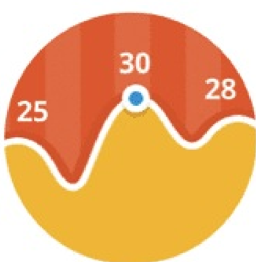
Analysis



Collect Data



Process Data



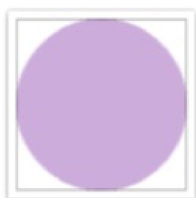
Write Report



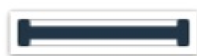
Dessimination



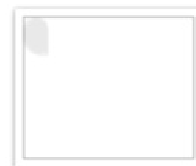
以打印机这个动画为例，我们需要一个纸张，一个连杆，还有打印机的基座：



bg.png



handlebar.png



paper.png



typewritter.png

有了这些基础元素，我们首先需要将各个元素按照静态的方式摆放整齐。首先需要定义合理的DOM结构：

```
<section class="typing">
    <div class="typewritter"></div>
```

```
    <div class="handlebar"></div>
    <div class="paper"></div>
</section>
```

然后就可以进行静态内容的摆放了：

```
.typing {
    background-color: #CBADDA;
    width: 220px;
    height: 220px;
    border: none;
    border-radius: 50%;
    position: relative;

    .typewritter {
        position: absolute;
        background-image: url('/images/typing/typewritter.png');
        background-size: cover;
        width: 144px;
        height: 107px;
        top: 120px;
        left: 38px;
        z-index: 3;
    }

    .handlebar {
        position: absolute;
        background-image: url('/images/typing/handlebar.png');
        background-size: cover;
        width: 169px;
        height: 28px;
        top: 108px;
        left: 26px;
        z-index: 1;
    }

    .paper {
        position: absolute;
        background-image: url('/images/typing/paper.png');
        background-size: cover;
        width: 127px;
        height: 106px;
        top: 50px;
        left: 40px;
        z-index:2;
    }
}
```

这个例子中的图片尺寸都是按照设计师提供的为准。通过将容器元素 `.typing` 的 `position` 属性设置为 `relative`，所有的子元素的 `position` 属性设置为 `absolute`，然后通过绝对定位来定位。我们只需要调整元素的top/left即可完成初步的布局：



动起来：

让连杆动起来看起来很容易，只需要保证他左右摆动（通过使用转换）即可：

```
@keyframes handlebar {
    0% {
        transform: translateX(0);
    }

    25% {
        transform: translateX(10px);
    }

    50% {
        transform: translateX(0);
    }

    75% {
        transform: translateX(-10px);
    }

    100% {
        transform: translateX(0);
    }
}
```

然后再快速的应用这个关键帧即可：

```
.handlebar {
    animation: handlebar .5s ease-in infinite;
```

```
}
```

可以看到，打印机的连杆动起来了，但是我们的帧分配的太过平均，我们可以植入一些值，使得动画更加真实：

```
80% {
    transform: translateX(0);
}
```

这个帧插入后，连杆从75%到80%会突然卡一下，这样看着会更加真实一些。

做完连杆之后，我们来做纸张的动画。纸张的基本动作很简单，就是上下移动：

```
@keyframes paper {
  0% {
      transform: translateY(0);
  }

  30% {
      transform: translateY(-10px);
  }

  50% {
      transform: translateY(-20px);
  }

  60% {
      transform: translateY(-30px);
  }

  80% {
      transform: translateY(-40px);
  }

  90% {
      transform: translateY(-45px);
  }

  100% {
      transform: translateY(0);
  }
}
```

然后应用这个关键帧给 `.paper` 即可：

```
.paper {
    animation: paper 2s ease-out infinite;
}
```

为了更加真实，让纸张有跳动感，我们可以插入一些帧，让纸张在完成向上的移动后，有一个短暂的（两个像素）回跳：

```
@keyframes paper {
  0% {
      transform: translateY(0);
  }

  30% {
      transform: translateY(-10px);
  }

  35% {
      transform: translateY(-8px);
  }

  50% {
      transform: translateY(-20px);
  }

  55% {
      transform: translateY(-18px);
  }

  60% {
      transform: translateY(-30px);
  }

  65% {
      transform: translateY(-28px);
  }

  80% {
      transform: translateY(-40px);
  }

  85% {
      transform: translateY(-28px);
  }

  90% {
      transform: translateY(-45px);
  }

  100% {
      transform: translateY(0);
  }
}
```

我们再来实现一个稍微复杂一些的动画效果：



首先还是将图层通过静态方式定位：

```
<section class="mag">
    <div class="magazine"></div>
    <div class="picture"></div>
    <div class="yellow-bar"></div>
    <div class="blue-bar"></div>
</section>
```

我们定义了4个元素，左边的图标，右边的图标，两个小的指示器。对应的CSS为：

```
.mag {
    background-color: #6A9ACE;
    width: 220px;
    height: 220px;
    border: none;
    border-radius: 50%;
    position: relative;
    overflow: hidden;

    .magazine {
        position: absolute;
        background-image: url('/images/dissemination-assets/magazine.png');
        background-size: cover;
        width: 139px;
        height: 157px;
        top: 36px;
        left: 30px;
    }
```

```
    .picture {
        position: absolute;
        background-image: url('/images/dissemination-assets/pic.png');
        background-size: cover;
        width: 94px;
        height: 88px;
        top: 94px;
        left: 108px;
    }

    .yellow-bar {
        position: absolute;
        background-image: url('/images/dissemination-assets/yellow.png');
        background-size: cover;
        width: 27px;
        height: 9px;
        top: 116px;
        left: 166px;
    }

    .blue-bar {
        position: absolute;
        background-image: url('/images/dissemination-assets/blue.png');
        background-size: cover;
        width: 25px;
        height: 14px;
        top: 154px;
        left: 164px;
    }
}
```

首先来看左边的图表，仔细观察GIF会发现，当图标移动到位置之后，还会向回弹一点。这个效果模拟了现实世界中的重力，显得更加真实。这个图表会略微停留一点时间，然后消失，仔细观察发现它运动的时间大约占所有时间的30%左右，因此我们需要它在30%前就完成所有的动作，然后停留剩余的70%时间。

```
@keyframes right-jump {
    0% {
        top: -50px;
        left: -150px;
        transform: rotate(0deg);
    }

    15% {
        transform: rotate(-20deg);
    }

    18% {
        transform: rotate(-10deg);
    }

    20% {
        transform: rotate(-3deg);
    }

    25% {
        top: 36px;
```

```
        left: 30px;
        transform: rotate(5deg);
    }

    30% {
        top: 36px;
        left: 30px;
        transform: rotate(0deg);
    }
    100% {
        display: none;
    }
}
```

开始时，元素位于一个不可见的位置，到达15%的时候，它会逆时针旋转20度，由于这时候元素还不可见（容器元素的限定了overflow为hidden），因此这里相当于重新设置了元素的初始状态。18%的时候逆时针旋转18度，相对于上一个状态，它事实上是顺时针旋转了2度。一直到25%，元素旋转了25度。然后紧接着在30%的时候，我们让元素旋转到0度，这相当于向回转。

这样，该元素就会相对较慢的旋转到指定位置，然后像刹车时的惯性一样弹回到最终位置。默认的，rotate是根据元素的中心为动画基准点的，我们需要以元素的左下角为基准点，因此需要设置 `transform-origin` 属性为 `bottom left` ：

```
.magazine {
    animation: right-jump 2s ease-in-out infinite;
    transform-origin: bottom left;
}
```

同样，右边稍小一点的图表也可以应用类似的动画效果：

```
@keyframes left-jump {
    0% {
        top: 100px;
        left: 260px;
        transform: rotate(0);
    }
    40% {
        top: 100px;
        left: 260px;
        transform: rotate(8deg);
    }
    44% {
        top: 95px;
        left: 230px;
        transform: rotate(6deg);
    }
    48% {
        top: 90px;
        left: 200px;
        transform: rotate(4deg);
    }
    52% {
        top: 85px;
```

```
        left: 170px;
        transform: rotate(2deg);
    }
    56% {
        top: 90px;
        left: 140px;
        transform: rotate(0deg);
    }
    60% {
        top: 96px;
        left: 106px;
        transform: rotate(-3deg);
    }
    65% {
        top: 94px;
        left: 108px;
        transform: rotate(0);
    }
    100% {
        top: 94px;
        left: 108px;
        display: none;
    }
}
```

最后，两个小图例是一个缩放的动画，但是时机需要正好在较小的图表就位之后。

```
@keyframes bar {
    0% {
        transform: scale(0);
    }

    80% {
        transform: scale(0);
    }

    90% {
        transform: scale(1);
    }

    95% {
        transform: scale(1);
    }

    100% {
        display: none;
    }
}
```

由于这两个图例出现的时机比较晚，我们需要将实际缩放的时机拖后，即，在80%的时候， `sacle` 的参数还是0，相当于在它的整个动画周期中，动画启动的时机会延后，这样它就可以和其他的元素动画同步起来了。

```
.yellow-bar {
```

```
    animation:  bar 2s ease infinite;
}

.blue-bar {
    animation:  bar 2s ease infinite;
}
```

移动端页面

附录：使用CSS框架

虽然我们可以通过完全手写的方式来完成页面的布局，过渡动画，配色等，但是有时候这些过程比较繁琐，特别是涉及到响应式设计之后，我们需要考虑如何为不同的屏幕尺寸配置不同的百分比。

使用CSS框架可以简化这些工作，特别是关于布局和响应式设计的部分。目前已经有很多的相关框架，比如应用最为广泛的Bootstrap，Foundation，Pure CSS等等。使用框架可以大大简化页面的开发进程。